Neurocomputing 72 (2009) 1160–1178

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Design of experiments on neural network's training for nonlinear time series forecasting

P.P. Balestrassi [a,*], E. Popova [b], A.P. Paiva [a], J.W. Marangon Lima [a]

[a] Federal University of Itajuba, Brazil
[b] University of Texas at Austin, USA

ARTICLE INFO

Article history:
Received 10 August 2007
Received in revised form
9 February 2008
Accepted 20 February 2008
Communicated by T. Heskes
Available online 29 February 2008

Keywords:
Design of Experiment
Artificial Neural Network
Nonlinear time series

ABSTRACT

In this study, the statistical methodology of Design of Experiments (DOE) was applied to better determine the parameters of an Artificial Neural Network (ANN) in a problem of nonlinear time series forecasting. Instead of the most common trial and error technique for the ANN's training, DOE was found to be a better methodology. The main motivation for this study was to forecast seasonal nonlinear time series—that is related to many real problems such as short-term electricity loads, daily prices and returns, water consumption, etc. A case study adopting this framework is presented for six time series representing the electricity load for industrial consumers of a production company in Brazil.

1. Introduction

DOE is considered one of the most important methodologies for researchers who deal with experiments in practical applications, with a huge amount of success stories. Nowadays, DOE resources are incorporated in many statistical software packages that ease calculation and interpretation of results [8]. Similarly, ANNs play an important role for problems of time series forecasting [19]. One often quoted drawback in using ANNs, however, is the optimization of the ANN's parameters. In general, the lengthy trial-and-error process is what most practitioners use for this optimization [63].

The main motivation for this work is to forecast nonlinear and seasonal time series—which is a real problem for many applications—using ANN. Usually daily, monthly, or yearly seasonality is inherent to several problems related to prices, returns, electricity load, water consumption, demand, etc. Nonlinear structures are also present in most cases. DOE is here used to estimate the parameters of an ANN through simulation.

There is a lack of literature on this specific issue and some questions are subject to investigation. Among others, the following questions will be addressed in this paper: *How to approach problems of nonlinear seasonal time series using ANNs*? *How to train ANNs for this kind of problem*? *How many factors are important to this approach*? *Are there interactions that should be considered*?

This paper describes in Section 2 the recent literature review of DOE on ANNs for problems of nonlinear time series forecasting. Section 3 presents numerical and graphical results for the overall procedure. Section 4 presents a case study for short-term electricity load problem using the DOE on ANN's framework. Section 5 states our main conclusions.

2. Background and literature review

2.1. Design of experiments for simulation

The process of training an ANN consists in changing the input parameters of a computational algorithm, running the algorithm, and checking the results. This can be referred as a simulation study for the ANN problem.

In spite of the amount of success stories related to industrial applications, DOE is not used as widely in simulation as it should be. Kleijnen et al. [34] points that the lack of use of DOE for simulation is due to some reasons: (i) Simulation analysts are not convinced of the benefits of DOE. (ii) DOE research is often found in specialty journals seldom read by simulation analysts. (iii) Most DOEs were originally developed for real-world experimentation rather than developed specifically for simulation settings. The aforementioned research also points the main benefits of experimental design on model development and simulation and predicts that the use of DOE is likely to become more substantial in this area:

> DOE can uncover detailed insight into the model's behavior, cause the modeling team to discuss in detail the implications

* Corresponding author. Tel.: +55 3591198924.
E-mail address: pedro@unifei.edu.br (P.P. Balestrassi).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.02.002

of various model assumptions, help frame questions when the analysts may not know ahead of time what questions should be asked, challenge or confirm expectations about the direction and relative importance of factor effects, and even uncover problems in the program logic … Design suited to a particular application is much better than trial and error or a simple, small design. Consequently, practitioners should be open to the notion that DOE is a useful and necessary part of analysis of complex simulation.

Translating the simulation terminology it could be said that an *input* or a *parameter* in simulation is referred to as a *factor* in DOE. Usually there are many more factors in simulation than in a real-world experiment. A factor can be either *qualitative* or *quantitative*. Each factor can be set to two or more values, called *factor levels*, typically coded numerically for analysis purposes. A scenario or *design* point is a combination of levels for all factors. In considering stochastic simulations *replicates* mean that different Pseudo-Random Numbers (PRNs) are used to simulate the same scenario. The nature of the data collection of scenarios is not random but sequential. Unless otherwise specified, it is assumed that replicates use nonoverlapping PRN streams, so outputs across replicates are *Independently Identically Distributed* (IID)—as most statistical methods assume. *Developing a basic understanding* in simulation is referred as *testing hypotheses about factor effects* in DOE [34].

Another important issue when using DOE for simulation is that the main goal here is not optimization. In using DOE the efforts are dedicated to find robust policies or decisions, rather than optimal policies. It is certainly true that finding the optimal policy for a simulated system is a hot topic, and many methods have been proposed. Fu [18] and Spall [50] discuss the current research and practice of optimization for simulation. These methods include heuristic search techniques—such as genetic algorithms, response surface methodology (RSM), simulated annealing, tabu search—and methods that analyze the simulation model to estimate gradients—such as perturbation analysis and score functions. The result of the "optimization" is conditioned on assumptions of specific (typically assumed independent) distributions and many input variables. The term "optimum" is problematic when the probability of all these assumptions holding in practice—even for a limited time—is effectively zero. In contrast, a robust design approach treats all these assumptions as additional factors when running the experiment. These are considered noise factors (rather than decision factors) because they are unknown or uncontrollable in the real-world environment. A robust system or policy works well across a range of noise conditions that might be experienced, so implementing a robust solution is much less likely to result in unanticipated results. This robust design philosophy is inspired by Taguchi [65], who uses simple designs to identify robust product configurations for Toyota.

2.2. Artificial Neural Networks for problems of time series forecasting

ANNs, first used in the fields of cognitive science and engineering, are universal and highly flexible function approximators. As cited by Tsay [56], ANNs are general and flexible tools for forecasting applications:

A popular topic in modern data analysis is ANN, which can be classified as a semiparametric method. As opposed to the model-based nonlinear methods, ANNs are data-driven approaches which can capture nonlinear data structures without prior assumption about the underlying relationship in a particular problem.

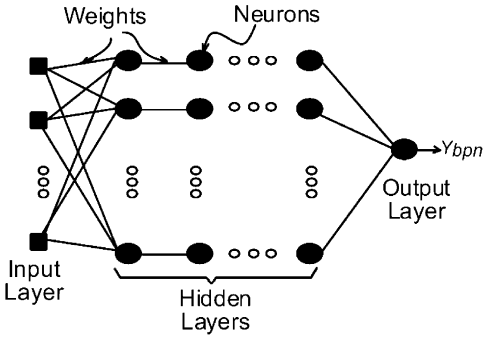

Fig. 1. Multilayer feedforward ANN structure.

Fig. 1 shows the ANN structure employed in the present study: A multilayer feedforward network trained with Backpropagation. The ANN has three types of layers, namely, the input layer, the output layer and the hidden layer, which is intermediate between the input and output layers. The number of hidden layers is usually 1 or 2. Each layer consists of neurons, and the neurons in two adjacent layers are fully connected with respective weights, while the neurons within the same layer are not connected. In this paper, the output layer has just a single neuron, which represents the one-step forecasting based on previous points.

Each neuron in the input layer is designated to an attribute in the data, and produces an output which is equal to the (scaled) value of the corresponding attribute. For each neuron in the hidden or output layer, the following input–output transformation is employed:

$$v = f\left(\sum_{h=1}^{H} w_h u_h + w_0\right),$$

where v is the output, H is the total number of neurons in the previous layer, u_h is the output of the hth neuron in the previous layer, w_h is the corresponding connection weight, w_0 is the bias (or intercept). f is the nonlinear transformation function (or activation function) also used in the output layer. The following transformation function, as example, is employed very often:

$$f(z) = \frac{2}{(1+e^{-z})} - 1.$$

When the ANN is trained using the Backpropagation algorithm the weights and biases are optimized. The objective function employed for optimization is the sum of the squares of the difference between a desirable output (y_{target}) and an estimated output (y_{bpn}).

Review of ANNs from statistical and econometric perspectives can be found in [11]. Today, ANNs are used in a variety of modeling and forecasting problems. Although many models commonly used in real problems are linear, the nature of most real data sets suggests that nonlinear problems are more appropriate for forecasting and accurately describing it. ANN plays an important role for this kind of forecasting.

The literature on ANN is enormous and its applications spread over many scientific areas with varying degrees of success. In the M-Competition [39], M2-Competition [40] and M3-Competition [38] many participants used ANNs. The main reason for this increased popularity of ANNs is that these models have been shown to be able to approximate almost any nonlinear function arbitrarily close. Hence, when applied to a time series which is characterized by truly nonlinear dynamic relationships, the ANN will detect these and provide a superior fit compared to linear models, without the need to construct a specific parametric nonlinear time series model.

Addressing the problem of time series forecasting, some important papers have considered ANNs as a promising methodology and addressed important issues. Franses and van Homelen [17] explore the ability of ANNs to capture nonlinearity as implied by SETAR, Markov-Switching and GARCH models. Kaastra and Boyd [28] provide an introductory guide—an eight-step procedure—in the design of a neural network for forecasting economic time series data. Bodyanskiy and Popov [66] present a special ANN approach to forecasting financial time series based on the presentation of the series as a combination of quasi-periodic components. Tseng et al. [57] proposes a hybrid forecasting model, which combines the seasonal time series ARIMA (SARIMA) and the neural network Backpropagation (BP) models, named there as SARIMABP. Karunasinghe and Liong [30] investigate the performance of ANNs as a global model over the widely used local models (local averaging technique and local polynomials technique) in chaotic time series. In the paper of Aitkenhead et al. [1], oil, stream water, and climatic variables, were measured hourly over several month periods in two situations in North-East (NE) Scotland, using data loggers and other measuring instruments. The data sets were used to train neural networks using three different methods, including a novel, biologically plausible system. BuHamra et al. [7] combine the Box–Jenkins (BJ) and the ANN approaches to model time series data of water consumption in Kuwait. Shi et al. [49] investigate nonlinear time series modeling using the general state-dependent autoregressive model. Niska et al. [43] model the air quality using ANN, a difficult task due to both their chaotic and nonlinear phenomenon and high dimensional sample space. Zhang [61] presents a hybrid methodology that combines both ARIMA model and ANNs to take advantage of the unique strength of ARIMA model and ANNs in linear and nonlinear modeling. Kim [32] uses support vector machines (SVMs) for the prediction of financial time series. This study applies SVM to predicting the stock price index. Ho et al. [23] show a comparative study of ANN and ARIMA modeling in time series prediction. BP and Recurrent ANN gives satisfactory performance compared to ARIMA. Kohzadi et al. [35] compare ARIMA and ANN price forecasting performance. Terasvirta et al. [54] examine the forecast accuracy of linear autoregressive, smooth transition autoregressive (STAR), and ANNs for 47 monthly macroeconomic variables of the G7 economies. Ghiassi et al. [19] present a dynamic neural network model for forecasting time series events that uses a different architecture than traditional models. Balkin and Ord [3] explain a method, called *Automated ANNs*, that is an attempt to develop an automatic procedure for selecting the architecture of an Artificial Neural Network for forecasting purposes. Cubiles-de-la-Vega et al. [67] propose a procedure for designing a multilayer perceptron for predicting time series. It is based on the generation, according to a set of rules emerging from an ARIMA model previously fitted, of a set of nonlinear forecasting models. Kalaitzakis et al. [29] present the development and application of advanced neural networks to face successfully the problem of the short-term electric load forecasting, using actual hourly load data from the power system of the island of Crete, in Greece. Qi and Zhang [46] expose problems of the commonly used information-based in-sample model selection criteria in selecting ANNs for financial time series forecasting. Zhang and Qi [62] investigate the problem of seasonality and show that limited empirical studies on seasonal time series forecasting with neural networks yield mixed results. In Chiang et al. [12], it was reported that the ANN proved to be superior to regression models when the data availability is limited, e.g., newly launched mutual funds which have limited historical data.

This research is partially motivated by the results presented in the following papers. Zhang [60], examining the capability of ANN for linear time series using both simulated and real data, states that ANN is quite competent in modeling and forecasting linear time series in a variety of situations and simple neural structures are often effective. Hwarng and Ang [27] and Hwarng [26] mainly motivated by linear time series forecasting addresses some ideas that are relevant for the present work: (i) "Backpropagation Neural Networks (BPNNs) generally performed well and consistently for time series corresponding to ARMA(p,q) structures, mainly when a particular noise level was considered during the network training". (ii) "Given the well notion that multilayer feedforward NN may act as a universal approximators, it is reasonable to expect that BPNNs can perform at least comparably on linear data. If so, one may find it convenient to apply BPNNs regardless of the nature of data especially when the functional form of data is unknown". Zhang et al. [63] present a comprehensible state of the art survey of ANN applications in time series forecasting for the past decade and the following points are here considered: (i) "Overall, ANNs give satisfactory performance in forecasting". (ii) "There are many factors that can affect the performance of ANNs. However, there are no systematic investigations of these issues. The shot-gun (trial and error) methodology for specific problems is typically adopted by most researchers, which is the primary reason for inconsistencies in the literature". (iii) "A considerable amount of research has been done in this area given the fast-growing nature of the literature".

2.3. Nonlinear time series

Linear time series methods have been used widely for the past two decades. Recently, however, there has been increasing interest in extending the classical framework of Box and Jenkins [5] to incorporate nonstandard properties, such as nonlinearity, non-Gaussianity, and heterogeneity. In this way, a great number of nonlinear models have been developed, such as the bilinear model of Granger and Anderson [20], the threshold autoregressive (TAR) model of Tong [55], the state-dependent model of Priestley [45], the Markov switching model of Hamilton [22], the functional-coefficient autoregressive model of Chen and Tsay [10], among many others. Although the properties of these models tend to overlap somewhat, each is able to capture a wide variety of nonlinear behavior. In most time series, however, this kind of modeling is even more complex due to some features like high frequency, daily and weekly seasonality, calendar effect on weekend and holidays, high volatility and presence of outliers. In particular, it has been shown that the ANN model is able to approximate any well-behaved nonlinear relationship to an arbitrary degree of accuracy, in much the same way that an ARMA model provides a good approximation of general linear relationships [9,24]. This is the so-called *universal approximation property* of ANNs. In short, feedforward ANN with a hidden layer can be seen as a way to parameterize a general continuous nonlinear function.

The problem one immediately faces when considering the use of nonlinear time series models is the vast, if not unlimited, number of possible models. Consider a univariate time series x_t which for simplicity is observed at equally spaced time points. We denote the observations by $\{x_t|t=1,\ldots,T\}$, where T is the sample size. A purely stochastic time series x_t is said to be linear if it can be written as

$$x_t = \mu + \sum_{i=0}^{\infty} \psi_i a_{t-i},$$

where μ is a constant, ψ_i are real numbers with $\psi_0 = 1$, and $\{a_t\}$ is a sequence of independent and identically distributed (IID) random variables with a well-defined distribution function. We assume

that the distribution of a_t is continuous and $E(a_t)=0$. In many cases, we further assume that $\mathrm{Var}(a_t)=\sigma_a^2$ or, even stronger, that a_t is Gaussian. If $\sigma_a^2\sum_{i=1}^{\infty}\psi_i^2<\infty$ then x_t is weakly stationary (i.e., the first two moments of x_t are time-invariant). The ARMA process is linear because it has an MA representation in the mentioned equation. Any stochastic process that does not satisfy the condition of this equation is said to be nonlinear. See [56] for more details. For previous and more general surveys on nonlinear time series models, the interested reader is referred to [55,21].

A natural approach to modeling time series with nonlinear models seems to define different *states of the world* or *regimes*, and to allow for the possibility that the dynamic behavior of variables depends on the regime that occurs at any given point in time [45]. By *state-dependent* dynamic behavior of a time series it is meant that certain properties of the time series, such as its mean, variance and/or autocorrelation, are different in different regimes. In particular, autocorrelations tend to be larger during periods of low volatility and smaller during periods of high volatility. The periods of low and high volatility can be interpreted as different regimes. Of course, the level of volatility in the future is not known with certainty. The best one can do is to make a sensible forecast of this level and, hence, of the regime that will occur in the future [36]. The state-dependent, or regime-switching considers that the regime is stochastic and not deterministic, witch is relevant for many time series. There are two main classes of regime-switching models: (i) Regimes determined by observable variables that include the Bilinear model, the TAR (Threshold Autoregressive) model and the SETAR (Self-Exciting Threshold Autoregressive) model and (ii) Regimes determined by unobservable variables that include the MSW (Markov-Switching) model. We restrict our attention to models that assume that in each of the regimes the dynamical behavior of the time series is modeled with an AR model. In other words, the time series is modeled with an AR model, where the autoregressive parameters are allowed to depend on the regime or state. Generalizations of the MA model to a regime-switching context have also been considered [13], but we abstain from discussing these models here.

Table 1 shows a collection of nonlinear time series implemented and simulated for the present study. In each case, ε_t:$N(0,1)$ is assumed to be IID. These eight time series models are chosen to represent a variety of problems that have different time series characteristics. For example, some of the series have pure autoregressive (AR) or pure moving average (MA) correlation structures while others have mixed AR and MA components. Similar models (with different lags) were explored by Zhang [60].

A typical graph of the mentioned models is shown in Fig. 2 where a STAR model with an autoregressive component and a lag 24 seasonality (typically for hourly seasonality) is enforced.

Table 1
A collection of nonlinear time series models

Model	Equation
Sign autoregressive (SAR)	$y_t=\mathrm{sign}(y_{t-12})+\varepsilon_t$, where $\mathrm{sign}(x)=1, 0, -1$ if $x>0$, $x=0$, $x<0$, respectively
Bilinear (BL1)	$y_t=0.8y_{t-1}\varepsilon_{t-1}+\varepsilon_t$
Bilinear (BL2)	$y_t=0.3y_{t-1}-0.4y_{t-24}+0.6y_{t-1}\varepsilon_{t-1}+\varepsilon_t$
Threshold autoregressive (TAR)	$y_t=0.7y_{t-1}+\varepsilon_t$ for $\lvert y_{t-1}\rvert\leqslant 1=-0.4y_{t-1}-\varepsilon_t$ for $\lvert y_{t-1}\rvert>1$
Nonlinear autoregressive (NAR)	$y_t=0.6y_{t-1}/(\lvert y_{t-24}\rvert+3)+\varepsilon_t$
Nonlinear moving average (NMA)	$y_t=\varepsilon_t-0.5\varepsilon_{t-24}+0.1\varepsilon_{t-2}+0.3\varepsilon_{t-1}\varepsilon_{t-2}-0.3\varepsilon_{t-2}^2$
Smooth transition autoregressive (STAR1)	$y_t=0.7y_{t-1}+0.7y_{t-24}+[1+\exp(-10y_{t-1})]^{-1}+\varepsilon_t$
Smooth transition autoregressive (STAR2)	

2.4. DOE on ANN's training for nonlinear time series forecasting

While the nonlinear models in Table 1 can be useful for a particular problem and data, they do not have a general appeal for other applications. The pre-specification of the model form restricts the usefulness of these parametric nonlinear models since there are too many possible nonlinear patterns. In fact, the formulation of an appropriate nonlinear model to a particular data set is a very difficult task compared to linear model building because "there are more possibilities, many more parameters and thus more mistakes can be made" [38]. Furthermore, one particular nonlinear specification may not be general enough to capture all nonlinearities in the data. As Diebold and Nason [14] pointed out, "the overwhelming variety of plausible candidate nonlinear models makes determination of a good approximation to the true data-generating process a difficult task and the seemingly large variety of parametric nonlinear models is in fact a very small subset of the class of plausible nonlinear data-generating process".

As described in Section 2.2, ANNs are natural candidates to forecast nonlinear time series. However, the large number of parameters that must be selected to develop a neural network have meant that the design process involves much trial and error. Traditional methods of studying one-factor-at-a-time may lead to unreliable and misleading results, and at times may give wrong conclusions. This is mainly what trial and error does. Statistically designed experiments perform more efficiently as they consider multiple factors simultaneously and can detect important interactions. Using DOE as a process of planning experiments enables the collection of appropriate data using the minimum number of experiments while acquiring the necessary technical information. Insight into the main effects, as well as interaction effects of factors including noise are useful in decision making to determine the control factors for further expenditure of resources.

A few papers have studied the use of ANNs, simulated through design of experiments, in a different context to the proposed in this paper and exploring different factors and levels. Khaw et al. [31] describe in this seminal paper an innovative application of the Taguchi method for the determination of the design parameters that include both the micro-structural and macro-structural aspects of a neural network. The feasibility of using this approach was demonstrated by optimizing the design parameters of a back-propagation neural network for determining operational policies for a manufacturing system. Results drawn from this research showed that the Taguchi method provides an effective means to enhance the performance of the neural network in terms of the speed for learning and the accuracy for recall. Kim and Yum

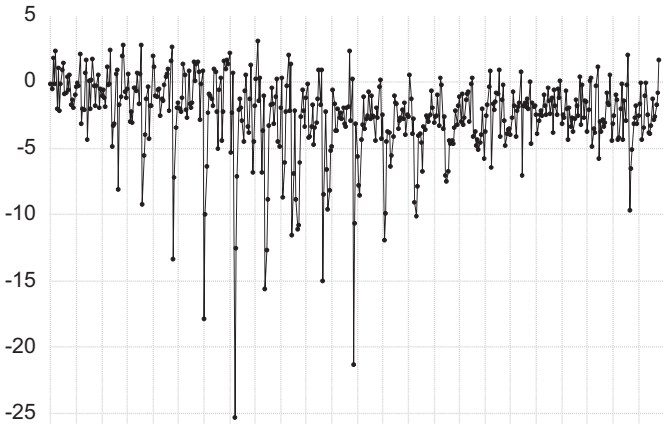

Fig. 2. A typical seasonal Smooth Transition Autoregressive (STAR2) time series.

[33] present a similar paper. Sukthomya and Tannock [52,53] use the same Taguchi experimental design idea to set the parameters of an ANN in a complex forming process. Lin and Tseng [37] also use the same Taguchi approach for a "Learning Vector Quantization" ANN on an application to bicycle derailleur systems. Enemuoh and El-Gizawy [16] describe a method for robust design of an ANN for prediction of delamination, damage width, and hole surface roughness during drilling in carbon fiber reinforced epoxy. The effects of number of neurons, hidden layers, activation function, and learning algorithm on the mean square error of model prediction are quantified. Using the aforementioned method, a robust ANN was developed that predicted process-induced damage with high accuracy. Zhang [60] and Zhang et al. [64] present an experimental evaluation of neural networks for linear and nonlinear time series forecasting. Three main ANN parameters are examined through a simulated computer experiment: input nodes, hidden nodes and sample size. The models used in these papers were similarly reproduced in the present work (see Table 2) for comparative purposes.

When compared with the previous papers, the present study could be mainly innovative in the following points:

- Instead of using the Taguchi approach of DOE, a mixed approach is examined;
- The number of ANN parameters is increased;
- The seasonality is included to mimic real nonlinear problems;
- Interactions among ANN parameters are permitted and evaluated.

3. Experimental design

In this section, the experimental design for the ANN's training is examined. First, some guidelines for industrial experiments will be addressed and some changes are recommended for the ANN's training context. The deployment and the results of the guideline are presented next.

3.1. Some guidelines

Coleman and Montgomery [68] present some guidelines for designing an experiment that in spite of been focused on industrial experiments can also be used for computer simulation:

(a) Recognition of and statement of the problem
(b) Choice of factors, levels, and ranges
(c) Selection of the response variable
(d) Choice of experimental design
(e) Performing the experiment
(f) Statistical analysis of the data
(g) Conclusions and recommendations

These guidelines are usually interactive and the structure is not rigid when applied in real experiments. Some steps are often done simultaneously or also in reverse order. Steps a, b and c are called *pre-experimental planning*. Some comments related to ANN's training are followed.

3.1.1. Recognition of and statement of the problem

This may seem an obvious point but in industrial experiments—and also in computer simulation—it is not simple to get the whole picture of a problem and usually a *team approach* is required. For this application, involving many different areas and expertise, where opinions are many times conflicting, a team approach is appropriated. Also, a clear *statement of the problem* contributes substantially for the problem solution. Recognition of the problem and its correct statement usually give focus to reach an objective.

3.1.2. Choice of factors, levels and ranges

In real-world experiments, only a small number of factors are typically varied. Indeed, it is impractical or impossible to attempt to control more than, say, 10 factors; many published experiments deal with fewer than 5. In contrast, a multitude of potential factors exists for simulation models used in practice.

Good programming avoids fixing the factors at specific numerical values within the code; instead, the code reads factor values so the program can be run for many combinations of values. Of course, the code should check whether these values are admissible; that is, do these combinations fall within the experimental domain? *Such a practice can automatically provide a list of potential factors*. Next, users should confirm whether they indeed wish to experiment with all these factors or whether they wish to fix some factors at nominal (or base) levels *a priori*. This type of coding helps *unfreeze* the mindset of users who would otherwise be inclined to focus on only a few factors.

In real-world experiments, the basic mindset is often that data should be taken simultaneously unless the design is specifically identified as a sequential design. When samples must be taken sequentially, the experiment is viewed as prone to validity problems. Analysts must therefore randomize the order of sampling to guard against time-related changes in the experimental environment (such as temperature, humidity, consumer confidence, and learning effects) and perform appropriate statistical tests to determine whether the results have been contaminated. Most simulation experiments are implemented sequentially even if they are not formally analyzed that way. If a small number of design points are explored, this implementation may involve the analysts manually changing factor levels [34].

The increase in computer speeds has caused some analysts to add more details to their simulation models. Different analysts might use different set of factors and levels.

3.1.3. Selection of the response variable

For industrial experiments the choice of a useful and practical response variable such as yield, load, cost, etc., involves a gage capability study and the error is in many cases evaluated by repeatability and reproducibility study. For computer simulation, whereas the initial condition can be blocked (usually setting the base of a random number) the response is usually an estimated variable that can be repeated without error by establishing the same initial conditions. The experimental choice is also, in a certain extent, much more complex than it is in a computer simulation and need to be evaluated *a priori*. Average, standard deviation, etc., are often used as response variable. Multiple responses are also usual in industrial designed experiments and the simultaneous optimization of several response variables involves desirability functions (Derringer and Suich [69]).

3.1.4. Choice of experimental design

The term *design* denotes a matrix where the columns represent the input factors and each row represents a combination of factor levels. Choice of design involves sample size, the run order of the experiment (that for computer simulation is usually irrelevant), and several restrictions to compose the final matrix generated as a worksheet output.

Some statistical software packages like Minitab, Statistica, SPSS, JMP, Matlab, among many others, are good programs that offers a library of classical designs. These designs are usually generated with coded levels and are chosen based on number of factors and levels, alias structure and resolution, amount of time

Table 2
Screening factors for the ANN's training on nonlinear time series forecasting

Factor	Symbol	Levels	Number of levels
ANN architecture	–	***MLP***, RBF, GRNN, ARTMAP, …	1
Number of hidden layers	HL	0 (rarely), ***1***, ***2*** or 3 (rarely)	2
Number of units per layer	UL	$\boldsymbol{k \times (N+1)}$, where N is the number of input and $k = 1, 1.5, 2$	3
Regression output function	OF	**Linear, Logistic-range**	2
Problem type/input mode	PT	**Univariate time series/Regression**	2
Predict X steps ahead	–	**1**,2, 3, …	1
Steps used to predict	SP	**12, 24**	2
Phase 1 training algorithm	P1	**Backpropagation, Quick propagation, Delta-bar-delta**	3
Phase 2 training algorithm	P2	**Conjugate gradient descent, Quasi-Newton, Levenberg–Marquardt**	3
Epochs	Ep	**100, 400**	2
Learning rate	LR	**0.1, 0.9**	2
Initialization method	IM	**Unif(0,1), *N*(0,1)**	2
Stopping conditions (Target error)	SC	**0, 0.1**	2
Minimum improvement in error for training/selection	ET	**0, −0.1**	2
Minimum improvement in error for number of epochs	EE	**1, 25, 50**	3
Prune units	PU	**No, with small fan-out weights (pruning threshold = 0.05)**	2
Prune input variables	PI	**No, with small fan-out weights (pruning threshold = 0.05), with low sensitivity after training (ratio = 1)**	3
Weight decay regularization—Phase 1	W1	No, **Decay factor = 0.0001, Decay factor = 0.001, Decay factor = 0.01**	3
Weigend weight decay regularization—Phase 2	W2	No, **Decay factor = 0.0001, Decay factor = 0.001, Decay factor = 0.01**	3
Backpropagation tuning (conditional to *Phase 1 training algorithm*)	BP	**4 runs for a L_4 Taguchi design with factors (A—Adjust learning rate and momentum each epoch, B—Shuffle presentation order of cases each epoch and C—Add Gaussian noise)**	4
Quick propagation tuning (conditional to *Phase 1 training algorithm*)	QP	**4 runs for a L_4 Taguchi design with 3 factors (A—Learning rate B—acceleration and C—Add Gaussian noise)**	4
Delta-bar-delta tuning (conditional to *Phase 1 training algorithm*)	DD	**4 runs for a L_4 Taguchi design with 2 factors (A—learning rate and B—Add Gaussian noise)**	4
Sample size	SS	$\boldsymbol{k \times 24}$**, where** $\boldsymbol{k = 7, 14, 21}$	3
Sampling method	SM	**Random**, Bootstrapping**, Cross-validated**	2

Levels in bold were chosen for the experimental design.

and resources to run the experiments, etc. One interesting new program, the WebDOE, helps users to design their experiment through an easy-to-use Web interface (Crary Group [70]).

Kleijnen et al. [34], using DOE for simulation, recommends a scheme based on the system complexity assumptions and states that it is better initially to focus on a large number of factors (respecting limitations of time, cost, etc.). In this way, analysts can look broadly across the factors in the simulation study. Sometimes intuition is wrong and needs to be challenged. With this approach, starting from minimal assumptions for an initial experiment, the analyst becomes open-minded for new assumptions and admits that little is known about the nature of the response. The analyst tends to reduce the initial data-collection effort making simplifying assumptions. This is chosen mainly when limitations are imposed. If runs are extremely time-consuming then analysts can reduce the computational effort by making assumptions about the nature of system. The best idea is to moving from minimal assumptions and to focus on the short list of factors selected after the initial experiment while holding the remaining factors to only a few configurations. Even after careful thought and planning, it is rare that the results from a single design are so comprehensive that the simulation model needs never be revisited. In practice, results from experiments often need to be modified, i.e., expanded or thrown out to obtain more detailed information on the simulation performance for a smaller region of the factor combinations.

3.1.5. Performing the experiment

To run the experiment in computer simulation is usually easier than in industrial experiments. If the simulation program is open-source, it is sometimes convenient to run the complete simulation as a batch process. Unfortunately, this is not what happens very often due to the programming skill needed and the software restriction always imposed. Also, this iterative experimentation is considered a mistake in industrial experiments. A design of a single large, yet comprehensive, matrix of experiments at the start of the study is not considered a good practice because the experimenter is not fully convinced of the levels and factors involved. Coleman and Montgomery [68] points out that the experimentation should be *sequentially* and, as a general rule, no more than about 25% of the available resources should be invested in the first experiment.

3.1.6. Statistical analysis of the data

Statistical methods here are not very elaborate if the previous guidelines are followed. Graphical methods, residual analysis and model adequacy play an important role in this phase. Statistical analysis add objectivity to the decision making process.

3.1.7. Conclusions and recommendations

If coupled with process knowledge and common sense, the statistical analysis can lead to sound conclusions and recommendations. Assuming a sequentially design experiment, follow-up runs and confirmation testing should also be performed.

3.2. Pre-experimental planning

In this section, the recognition of and statement of the problem, the selection of the response variable and the choice of factors, levels, and ranges will be discussed.

Related to the ANN's training, the following statement of the problem was considered appropriated, using a teamwork approach:

> When predicting nonlinear time series using ANN, trial and error is the shotgun and time consuming methodology often used. The problem here is to establish a well-structured methodology do estimate the parameters of such ANN.

Several error measures like Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), Median Absolute Percentage Error (MdAPE), etc., have been used as performance measures of time series forecasting. The interested reader could check Armstrong and Collopy [71] for a better discussion on this topic. It should be noted that there is no uniformly accepted forecasting error measure. In this work, the traditional MAPE will be used as response variable, defined as

$$\text{MAPE}\,(\%) = \frac{1}{T}\sum_{t=1}^{T}\frac{|y_t - \hat{y}_t|}{y_t} \times 100,$$

where y_t is the actual observation at time t, $\hat{y}_t$ is the predicted value, and T is the number of predictions.

Several factors have been considered in the literature when training ANNs. Table 2 presents the screening factors considered for problems of nonlinear time series forecasting using the software Statistica (with Neural Network toolbox) [51]. Details about the factors are described next.

- *ANN architecture*. ANNs are nonlinear modeling algorithms. Examples of ANN for nonlinear time series are Multilayer Perceptrons (MLP), Radial Basis Function (RBF), Generalized Regression Neural Network (GRNN), Support Vector Machine (SVM), among many others. MLPs are one of the most popular network types, the only one considered in this work, and in many problem domains seem to offer the best possible performance.
- *Number of hidden layers*. A hidden layer is a group of neurons that have a specific function and are processed as a whole. Theoretical results prescribe that an MLP with one hidden layer (three layer perceptron) is capable of approximating any continuous function [25].
- *Number of units per layer*. Hidden nodes are used to capture the nonlinear structures in a time series. Here, it will be used an amount between the number of input and its double ($k(N+1)$, where N is the number of input and $k = 1, 1.5, 2$).
- *Regression output function* (*activation function*). All neural networks take numeric input and produce numeric output. The transfer function of a unit is typically chosen so that it can accept input in any range, and produces output in a strictly limited range (it has a squashing effect) [4].
- *Problem type/input mode*. The way the time series is presented as input to the ANN is always considered fundamental. Here, two ways of presenting the time series to the ANN will be tested.
  (a) As a problem of *univariate time series* the lagged information for the autoregressive process is preserved. In this way, the input will be a sample of the own time series $[y_{t-1}, y_{t-2}, \ldots, y_{t-23}, y_{t-24}]$ considering a 24 lag, and the supervised response will be the one-step-ahead observation $[y_t]$.
  (b) As a problem of *regression*, temporal explanatory (dummy) variables will be used. In this way, the input will be a set of dummy values (0s and 1s) plus the time series observation $[y_{t-1}, y_{t-2}, \ldots, y_{t-x}, d_1, d_2, \ldots, d_k,]$, where x represents some last observations and d_k is a binary transformation for the seasonality or other endogenous variables—e.g., the hour 12 could be transformed into the dummy number 1100. Procedures of data mining are usually useful to pre-process the original time series for obtaining special features like variance, autocovariance, average, etc., as input for the ANNs. This study assumes the number of inputs in case (a) as well as in case (b) as the same so we can compare the methodologies. Due to that, we should know in advance the most important seasonality to define the lagged variables. Graphs like Autocorrelation function (ACF) or density functions (generated by Fast Fourier Transform—FFT) are available resources in many packages that could help define the seasonality. As a naïve time series, the ACF and the density function for the STAR2 model (as first presented in Fig. 2) shows clearly the seasonality in Fig. 3.

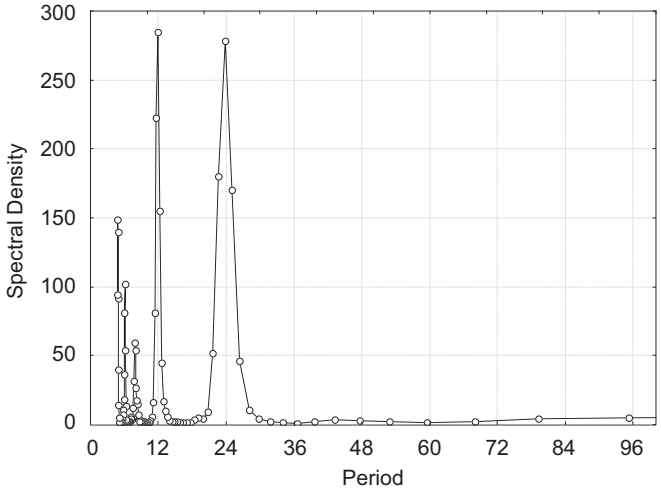

Fig. 3. Spectral density for the STAR2 model.

- *Predict X steps ahead*. X represents the number of steps ahead of the lagged input values that the predicted output lies. In this case, due to small synthetic time series and considering the error propagation throughout the steps prediction, just one step ahead will be used. The output of the network can be combined with previous input values, shifted one time step, and repeated predictions made. Since the runtime is mainly dependent on the minimum error to be reached and this error is not linear, it is not correct to say that predicting two steps ahead doubles the runtime of predicting one step ahead. Since the second prediction aggregate also the error of the first prediction, the second runtime cost tends to be greater than the first one. The prediction for X (greater than 1) steps ahead with the consequent shift of X time steps on the input, presents two drawbacks: the runtime cost and the MAPE are increased.
- *Steps used to predict*. The number of ANN's input are considered here as 24 and 12 either as a univariate time series mode or as a regression mode.
  (a) For the time series input mode, the level 24 corresponds to a complete period considering seasonality. The level of 12 corresponds to a composition between the last 8 points of the series plus 4 correspondents' statistics of the input series (average, standard deviation, autocorrelation for lag 1 and autocorrelation for lag 2).
  (b) For the regression input mode, the level of 24 corresponds to a composition of 20 values of the series plus 4 dummy variables related to the last point. The level of 12 corresponds to a composition between the last 4 points of the series plus 4 correspondents' statistics of the input series (average, standard deviation, autocorrelation for lag 1 and autocorrelation for lag 2) plus 4 dummy variables related to the last point.

- *Phase 1 training algorithm*. This factor is related to the training algorithm for the MLP in a first stage and assumes three levels.
  (a) *Backpropagation*. A simple algorithm with a large number of tuning parameters, often slow terminal convergence, but good initial convergence.
  (b) *Quick propagation*. An older algorithm with comparable performance to Backpropagation in most circumstances, although it seems to perform noticeably better on some problems.
  (c) *Delta-bar-delta*. Another variation on Backpropagation, which occasionally seems to have better performance.
- *Phase 2 training algorithm*. This factor is related to the training algorithm for the MLP in a second stage and assumes three levels.
  (a) *Conjugate gradient descent*. A good generic algorithm with generally fast convergence.
  (b) *Quasi-Newton*. A powerful second-order training algorithm with very fast convergence but high memory requirements.
  (c) *Levenberg–Marquardt*. An extremely fast algorithm in the right circumstances (i.e., low-noise regression problems with the standard sum-squared error function).
- *Epochs*. An epoch is the presentation of the entire training set to the neural network in a given phase. Increasing this number will likely improve the accuracy of the model, but at the cost of time, and decreasing this number will likely decrease the accuracy, but take less time. Directly related to the time series sample size this value will be adopted here at the levels of 100 and 400.
- *Learning rate*. A value between 0 and 1 that represents a tuning variable for the training algorithms of Backpropagation, Quick propagation and Delta-bar-delta. Lower learning rates require more training iterations. A higher learning rate allows the network to converge more rapidly, however the chances of a non-optimal solution are greater. The levels chosen here were 0.0 and 0.1.
- *Initialization method*. This factor defines how the weights should be initialized at the beginning of training and assumes two levels:
  (a) *Random uniform*. The weights are initialized to a uniformly distributed random value, within a range whose minimum and maximum values are given. In this case, minimum and maximum are 0 and 1.
  (b) *Random Gaussian*. The weights are initialized to a normally distributed random value, within a range whose mean and standard deviation are given. In this case, $N(0,1)$ was adopted.
- *Stopping conditions* (*Target error*). If the error on the training or selection test drops below the given target values, the network is considered to have trained sufficiently well, and training is terminated. The error never drops to zero or below, so the default value of zero is equivalent to not having a target error.
- *Minimum improvement in error for training/selection*. This factor represents the minimum improvement (drop) in error that must be made; if the rate of improvement drops below this level, training is terminated. The default value of zero implies that training will be terminated if the error deteriorates. One can also specify a negative improvement rate, which is equivalent to giving a maximum rate of deterioration that will be tolerated. The improvement is measured across a number of epochs, called the "window" (see below).
- *Minimum improvement in error for number of epochs*. Specifies the number of epochs across which improvement is measured. Some algorithms, including Backpropagation, demonstrate noise on the training and selection errors, and all the algorithms may show noise in the selection error. It is therefore not usually a good idea to halt training on the basis of a failure to achieve the desired improvement in error rate over a single epoch. The window specifies a number of epochs over which the error rates are monitored for improvement. Training is only halted if the error fails to improve for that many epochs. If the window is zero, the minimum improvement threshold is not used at all. The variable levels of 1, 25 and 50 define the window size in determining minimum improvement. The adopted values were considered taking into account the time series seasonality.
- *Prune units with small fan-out weights*. A neuron with small magnitude fan-out weights (i.e., weights leading to the next level) makes little contribution to the activations of the next layer and can be pruned, leading to a compact, faster network with equivalent performance. This option is particularly useful in conjunction with *Weigend weight decay* that encourages the development of small weights precisely so that they can be pruned. Hidden units with small fan-out weights should be pruned. If a unit's fan-out weights have smaller magnitude than a threshold, it is a candidate for pruning.
- *Prune input*.
  (a) … *with small fan-out weights*. Each input has one or more associated input layer neurons (more than one for some nominal inputs, and for time series networks), and the fan out on all these input neurons must be less than the threshold for the input to be pruned.
  (b) … *with low sensitivity after training*. A sensitivity analysis is run after the network is trained, and input with training and selection sensitivity ratios below a threshold value are pruned. An input with a sensitivity pruning ratio threshold of 1.0 makes no contribution to the network's decision, and can be pruned without any detriment. An input with sensitivity below 1.0 actually damages network performance, and should definitely be pruned (perhaps surprisingly, inputs with sensitivity below 1.0 on the selection data are not an uncommon occurrence, a by-product of over-learning).
- *Weigend weight decay regularization—Phase 1*. Weight decay can be applied separately to the two phases of a two-phase algorithm. In the phase 1 the algorithms of Backpropagation, Quick propagation, Delta-bar-delta were considered. This option encourages the development of smaller weights, which tends to reduce the problem of over-fitting, thereby potentially improving generalization performance of the network, and also allowing you to prune the network. Weight decay works by modifying the network's error function to penalize large weights—the result is an error function that compromises between performance and weight size. Consequently, too large a weight decay term may damage network performance unacceptably, and experimentation is generally needed to determine an appropriate weight decay factor for a particular problem domain. A study of decay factors and weight pruning shows that the number of inputs and units pruned is approximately proportional to the logarithm of the decay factor, and this should be borne in mind when altering the decay factor. For example, if you use a decay factor of 0.001 and this has insufficient effect on the weights, you might try 0.01 next, rather than 0.002; conversely, if the weights were over-adjusted resulting in too much pruning, try 0.0001. There is also a secondary factor in weight decay, which is usually left at the default value of 1.0.
- *Weigend weight decay regularization—Phase 2*. In this phase, the algorithms of Conjugate gradient descent, Quasi-Newton, and Levenberg–Marquardt were considered.
- *Backpropagation tuning*:
  (a) *Adjust learning rate and momentum each epoch*. Usually Backpropagation uses a fixed learning rate and momentum throughout training. Some authors, however, recommend altering these rates on each epoch (specifically, by reducing

the learning rate—this is often counter-balanced by increasing momentum). A higher learning rate may converge more quickly, but may also exhibit greater instability. Values of 0.1 or lower are reasonably conservative—higher rates are tolerable on some problems, but not on all (especially on regression problems, where a higher rate may actually cause catastrophic divergence of the weights). Momentum is used to compensate for slow convergence if weight adjustments are consistently in one direction—the adjustment "picks up speed". Momentum usually increases the speed of convergence of Backpropagation considerably, and a higher rate can allow you to decrease the learning rate to increase stability without sacrificing much in the way of convergence speed.

(b) *Shuffle presentation order of cases each epoch*. In this case, the Backpropagation algorithm adjusts the weights of the network as each training case is presented (rather than the batch approach, which calculates an average adjustment across all training cases, and applies a single adjustment at the end of the epoch). If the shuffle option is checked, the order of presentation is adjusted each epoch. This makes the algorithm somewhat less prone to stick in local minima, and partially accounts for Backpropagation's greater robustness than the more advanced second-order training algorithms in this respect.

(c) *Add Gaussian noise*. Gaussian noise of the given deviation is added to the target output value on each case. This is another regularization technique, which can reduce the tendency of the network to overfit. The best level of noise is problem dependent, and must be determined by experimentation. The standard deviation of the Gaussian noise added to the target output during training in this case is 0.10.

- *Quick propagation tuning*:
  (a) *Learning rate*. Specify the initial learning rate, applied in the first epoch; subsequently, the quick propagation algorithm determines weight changes independently for each weight.
  (b) *Acceleration.* Specify the maximum rate of geometric increase in the weight change, which is permitted. For example, an acceleration of two will permit the weight change to no more than double on each epoch. This prevents numerical difficulties otherwise caused by non-concave error surfaces.
  (c) *Add Gaussian noise*. Same as for Backpropagation.
- *Delta-bar-delta tuning*:
  (a) *Learning rate.* Specify the initial learning rate used for all weights on the first epoch. Subsequently, each weight develops its own learning rate. Specify the linear increment added to a weight's learning rate if the slope remains in a consistent direction. Specify the geometric decay factor used to reduce a weight's learning rate if the slope changes direction. Specify the smoothing coefficient used to update the bar-Delta smoothed gradient. It must lie in the range (0,1). If the smoothing factor is high, the bar-Delta value is updated only slowly to take into account changes in gradient. On a noisy error surface this allows the algorithm to maintain a high learning rate consistent with the underlying gradient; however, it may also lead to overshoot of minima, especially on an already-smooth error surface.
  (b) *Add Gaussian noise*. Same as for Backpropagation.
- *Sample size*. The set of time series will be represented having an hourly based seasonality corresponding to 1 week (168 points), 2 weeks and 4 weeks (that is approximately a month). This is a tentative to resemble practical problems. The seed for the random error was recorded for future benchmarking purposes.
- *Sampling method*:
  (a) *Random resampling*. In the random (Monte Carlo) resampling method the subsets are randomly sampled from the available cases. Each available case is assigned to one of the three subsets (training, selection or test) using the proportion 2:1:1.
  (b) *Cross-validated resampling*. The cross validation is *N*-fold, where *N* is the number of samples taken. The available data is divided into *N* parts, and one part is assigned to the test set on each sample. The remainder is divided among the training and selection subsets, and you can specify how many are put in each. Many authors, when using cross validation, do not use a selection set at all, on the basis that any bias contributed by a particular network can be compensated for by averaging predictions across the networks in an ensemble. However, it is probably still advisable to take some steps to alleviate over-learning, which might include use of a selection set, weight decay, or stopping conditions determined experimentally to be reasonable for the problem domain.

3.3. Designs and results

In this section, the guidelines related to the choice of experimental design and its statistical results will be discussed.

Considering the 24 factors in Table 2 it is interesting to evaluate the complexity of running the entire combinatorial possibilities while simulating the ANNs. In a case where only two levels for each factor are considered to run the complete simulation (varying one factor at a time), the number of runs would be 2^{24}. Taking into account 10 min as a rough time for each simulation in a fast computer, the time required would be related to centuries for all the combinations. It is unlikely that trial and error, the most used method, find the best combination for the ANN's training. Fortunately, a great number of designs are available nowadays to deal with screening procedures.

Some potential designs and strategies are suitable for the first screening phase where the goals are to identify those factors that may affect the performance the most, screen out the irrelevant variables and establish the cause and effect relationships. The screening method used depends on the number of factors that need to be screened. Fractional factorial designs are amongst the most widely used types of design in industry (Myers and Montgomery [72]). However, those are not practical when the number of factors exceeds 20 and there are more than two levels. Other screening methods for large numbers of variables include (i) Group Screening Design (Kleijnen [73]) with some drawback related to group formation; (ii) Sequential Bifurcation (Trocine and Malone [74]), with a drawback of being limited to quantitative variables; (iii) the Iterated Fractional Designs (Andres and Hajas [75]) with a drawback of not being able to evaluate interaction, and (iv) the Trocine Screening Procedure (Trocine [76]) with a drawback of only being able to evaluate three to four critical factors. In general, these methods are useful and much better than the traditional trial and error strategy.

3.3.1. Design 1: Taguchi screening design for 19 factors

In this work, a Taguchi approach was considered as screening design due to the factor's structure presented in Table 2, with 11 two-level factors, 8 three-level factors and 3 conditional four-level factors. Out of 24 factors, 2 of them (*ANN architecture* and *Predict X steps ahead*) were established as a constraint and excluded from the design due to the following reasons: (i) The ANN architecture has to be defined *a priori* so the factors could be the same. Using a

Table 3
Taguchi crossed array

	Inner Array (L_{36})																		Outer Array (L_4)			
	11 two-level factors											8 three-level factors							y_1	y_2	y_3	y_4
																		BP	1	1	2	2
																		QP	1	2	1	2
																		DD	1	2	2	1
	HL	OF	PT	SP	Ep	LR	IM	SC	ET	PU	SM	UL	P1	P2	EE	PI	W1	W2	SS	MAPE			
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.056	0.066	0.071	0.073
2	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	0.073	0.078	0.073	0.075
3	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	0.084	0.076	0.074	0.085
4	1	1	1	1	1	2	2	2	2	2	2	1	1	1	1	2	2	2	2	0.093	0.100	0.102	0.096
5	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3	3	0.092	0.096	0.091	0.097
6	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	1	1	1	1	0.103	0.095	0.103	0.102
7	1	1	2	2	2	1	1	1	2	2	2	1	1	2	3	1	2	3	3	0.093	0.085	0.095	0.094
8	1	1	2	2	2	1	1	1	2	2	2	2	2	3	1	2	3	1	1	0.088	0.098	0.103	0.101
9	1	1	2	2	2	1	1	1	2	2	2	3	3	1	2	3	1	2	2	0.150	0.147	0.144	0.142
10	1	2	1	2	2	1	2	2	1	1	2	1	1	3	2	1	3	2	3	0.041	0.039	0.046	0.044
11	1	2	1	2	2	1	2	2	1	1	2	2	2	1	3	2	1	3	1	0.090	0.093	0.091	0.097
12	1	2	1	2	2	1	2	2	1	1	2	3	3	2	1	3	2	1	2	0.091	0.093	0.092	0.097
13	1	2	2	1	2	2	1	2	1	2	1	1	2	3	1	3	2	1	3	0.075	0.071	0.063	0.067
14	1	2	2	1	2	2	1	2	1	2	1	2	3	1	2	1	3	2	1	0.131	0.123	0.121	0.123
15	1	2	2	1	2	2	1	2	1	2	1	3	1	2	3	2	1	3	2	0.084	0.093	0.098	0.086
16	1	2	2	2	1	2	2	1	2	1	1	1	2	3	2	1	1	3	2	0.107	0.095	0.097	0.093
17	1	2	2	2	1	2	2	1	2	1	1	2	3	1	3	2	2	1	3	0.132	0.124	0.128	0.129
18	1	2	2	2	1	2	2	1	2	1	1	3	1	2	1	3	3	2	1	0.117	0.105	0.103	0.107
19	2	1	2	2	1	1	2	2	1	2	1	1	2	1	3	3	3	1	2	0.124	0.131	0.137	0.125
20	2	1	2	2	1	1	2	2	1	2	1	2	3	2	1	1	1	2	3	0.114	0.120	0.116	0.115
21	2	1	2	2	1	1	2	2	1	2	1	3	1	3	2	2	2	3	1	0.094	0.099	0.100	0.100
22	2	1	2	1	2	2	2	1	1	1	2	1	2	2	3	3	1	2	1	0.103	0.106	0.112	0.119
23	2	1	2	1	2	2	2	1	1	1	2	2	3	3	1	1	2	3	2	0.108	0.117	0.116	0.117
24	2	1	2	1	2	2	2	1	1	1	2	3	1	1	2	2	3	1	3	0.115	0.107	0.105	0.110
25	2	1	1	2	2	2	1	2	2	1	1	1	3	2	1	2	3	3	1	0.109	0.117	0.113	0.111
26	2	1	1	2	2	2	1	2	2	1	1	2	1	3	2	3	1	1	2	0.063	0.065	0.066	0.062
27	2	1	1	2	2	2	1	2	2	1	1	3	2	1	3	1	2	2	3	0.111	0.111	0.115	0.115
28	2	2	2	1	1	1	1	2	2	1	2	1	3	2	2	2	1	1	3	0.102	0.111	0.113	0.118
29	2	2	2	1	1	1	1	2	2	1	2	2	1	3	3	3	2	2	1	0.088	0.081	0.091	0.078
30	2	2	2	1	1	1	1	2	2	1	2	3	2	1	1	1	3	3	2	0.149	0.136	0.135	0.129
31	2	2	1	2	1	2	1	1	1	2	2	1	3	3	3	2	3	2	2	0.096	0.103	0.104	0.105
32	2	2	1	2	1	2	1	1	1	2	2	2	1	1	1	3	1	3	3	0.087	0.088	0.076	0.085
33	2	2	1	2	1	2	1	1	1	2	2	3	2	2	2	1	2	1	1	0.098	0.099	0.105	0.099
34	2	2	1	1	2	1	2	1	2	2	1	1	3	1	2	3	2	3	1	0.099	0.107	0.107	0.090
35	2	2	1	1	2	1	2	1	2	2	1	2	1	2	3	1	3	1	2	0.063	0.056	0.062	0.067
36	2	2	1	1	2	1	2	1	2	2	1	3	2	3	1	2	1	2	3	0.067	0.070	0.055	0.064

different ANN would imply in different factors and levels that does not make sense in this case. For instance, only MLP was here evaluated. (ii) The prediction horizon was defined for a single step ahead, which simplifies dramatically the comparison results. Just increasing this interval does not add any major contribution to the ANN optimization and to the research objective.

The Taguchi design with its structure of inner/outer array allows a good fitting for the factors in Table 2 as presented in Table 3. In this screening phase, the two and three-level factors are used as inner array in a L_{36} Taguchi design. The conditional factors (*BP*, *QP* and *DD*) are related to the tuning of the factor P1 (*Phase 1 training algorithm*) for the MLP and will be used here as outer array structure. This means that when a level (*Back-propagation*, *Quick Propagation or Delta-bar-Delta*) for the factor P1 is chosen, a specific tuning will be used for each level. A L_4 Taguchi design for the tuning $y_1 \ldots y_4$ was then chosen. This design is based on the levels (A, B and C of the tuning parameters of the factors *BP*, *QP* and *DD* according to Table 2. The response variable is the metric MAPE and the outer array structure is conditional to the above-mentioned level of the factor P1. When P1 assume, for example, level 1 (that means *Backpropagation*) the tuning will setup A (*Adjust learning rate and momentum each epoch*), B (*Shuffle presentation order of cases each epoch*) and C (*Add Gaussian noise*) as Yes (1) or No (2) according to the L_4 Taguchi design. Each run of L_{36} will be repeated 4 times through y_1 to y_4. The idea here was to use a design with minimal number of runs. In this case the L_4 Taguchi design (with 4 runs) is similar to a 2^{3-1} classical design (with also 4 runs). Both designs can be considered as of resolution III, were the alias structure is similar. For a high order design, like the full factorial design, the number of runs would be doubled (and so the number of simulations). In this case, in spite of the gain of resolution of a higher order design, the runtime cost is decisive. Since the L_4 Taguchi design is used only on the screening phase, and only for conditional factors, the confounding effects (due to the DOE resolution) could be solved in the next designs. The following experimental design was obtained and the MAPE, considering 24 or 12 points out of training sample, was then calculated.

Taguchi recommends that the mean response for each run in the inner array and also the signal-to-noise ratio (*SN*) need to be evaluated. The *SN* here can be computed using the following standard smaller-the-better function:

$$SN = -10 \log\left(\frac{1}{n}\sum_{i=1}^{n} y_i^2\right) \quad \text{with } n = 4.$$

The *SN* is expressed on a decibel scale and at this screening level, the Taguchi strategy of analysis is just to *pick the winner*. The

best level values for computed *SNs* are always the greatest. Values for the mean depend on the problem type and here MAPE is desirable as smaller as possible. Taguchi advocates claim that the use of *SN* ratio generally eliminates the need for examining specific interactions between the inner array and the outer array factors. By observing both the mean and the *SN* ratio, as shown in Fig. 4, it is easy to pick the factors level that result at the same time in smaller values of mean and greater values of *SNs*. Borderline values will be left for future designs where interactions could be better evaluated.

From this screening analysis, some results were established, considering the MAPE mean and the *SN* ratio:

(a) The factors IM, SC, PU, EE and PI were considered not significant and further designs could either eliminate or define them as a noise factors. These are the factors which effects were constant for the factor's level.
(b) The factors PT, P1 and P2 were considered significant and their levels were established for future designs. These are the factors which effects have resulted in both greater *SN* and smaller mean, that means P1 (*level 1 = Backpropagation*), P2 (*level 3 = Levenberg–Marquardt*) and PT (*level 1 = Univariate time series*).
(c) The factors HL, SM, UL, W1, W2, SS, OF, SP, Ep, LR and ET were considered borderlines because it was not so clear to eliminate or to select the factor's level. Here, further investigation is needed.
(d) Due to clear similarities in terms of *SN* as well as in terms of mean, the three-level borderline factors (UL, W1, W2 and SS) were then defined as two-level factors.
(e) For the noise variables, statistical *t*-tests have not rejected the null hypothesis of equal means. In this way the authors were quite comfortable in neglecting the noise variables in further designs.
(f) One interesting finding is related to the number of inputs pruned and the input mode. For the 24 points for a problem of univariate time series, the number of inputs pruned is small for most cases (the average was 3). This means that all the points are important to establish the prediction. As a problem of regression, were pre-processing was imposed to the time series input, the pruned number was statistically greater than for the univariate time series mode (the average number was 7). This means that some pre-processing techniques have not added any contribution for the neural network efficiency.

3.3.2. *Design 2:* 2_{III}^{11-7} *fractional factorial design for 11 factors*

For the 11 two-level remaining factors a resolution III Plackett–Burman fractional factorial design or a L_{12} Taguchi design could be used with potentially same results and using the minimum amount of 12 runs. Some close options could be the L_{16} Taguchi design or the resolution III Fractional Factorial 2_{III}^{11-7}design, both with 16 runs. Using both Plackett–Burman and Taguchi designs, the interaction analysis is often difficult to interpret in practice. If the choice is between a geometric 2_{III}^{11-7} design with 16 runs or a 12-run Plackett–Burman design that may have to be folded over (thereby requiring 24 runs) the geometric design may turn out to be a better choice (Montgomery, 1993). The geometric fractional factorial 2_{III}^{11-7} design (in coded units) was then used here and the experimental results are shown in

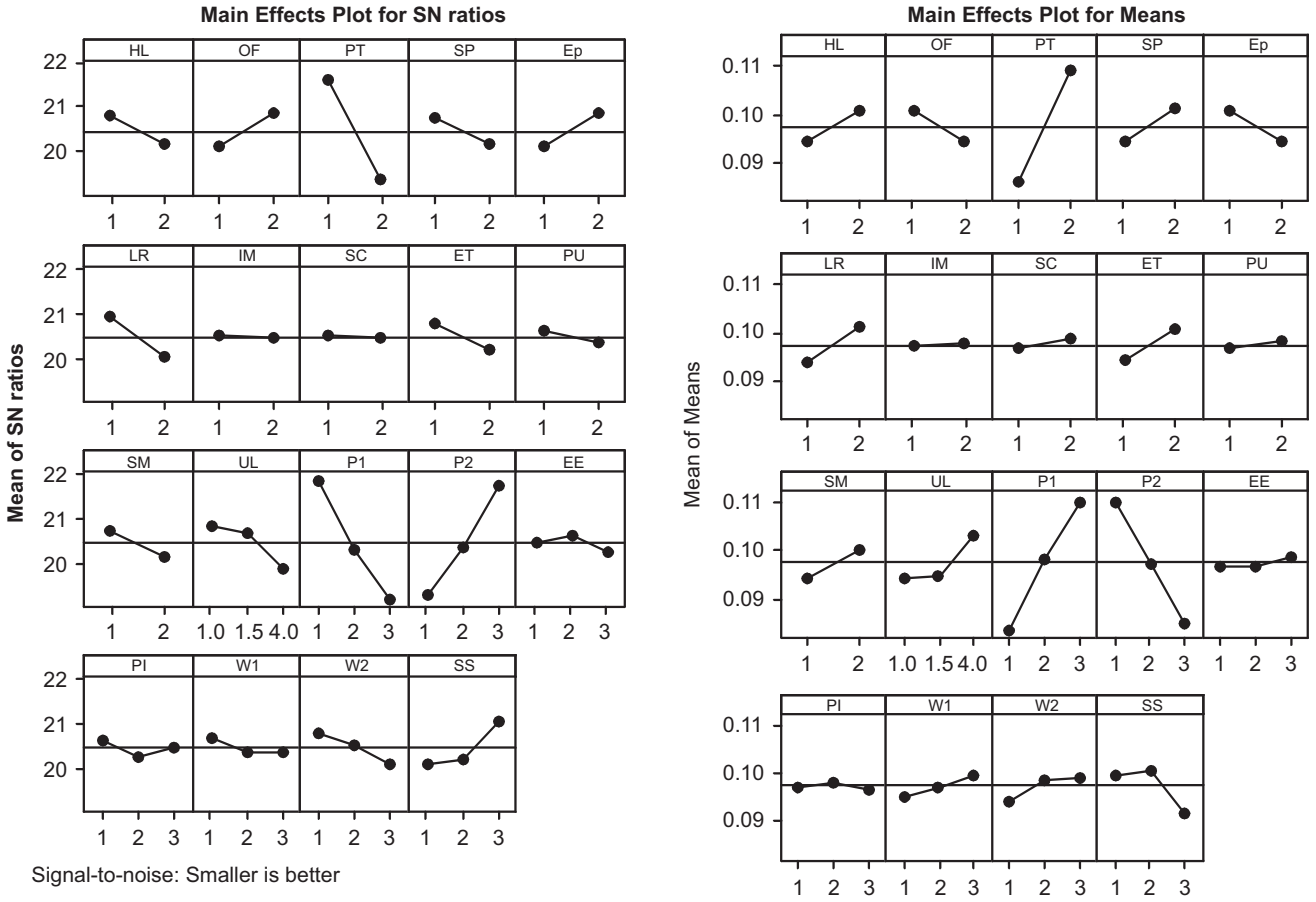

Fig. 4. Mean and *SN* for the Taguchi design (STAR2 model).

Table 4. MAPE 1, MAPE 2 and MAPE 3 consider the noise variables at their lowest, middle and upper levels, respectively.

Fig. 5 presents the Pareto chart and the main effects plot for the factors in Table 4.

For this second screening analysis, the results were:

(a) The factors ET and W1 were considered not significant and further designs could either eliminate or define them as a noise factor. These are factors which effects were constant for the factor's level.

(b) The factor LR was considered significant and its level could be established for future designs. This is a factor which effect has resulted in smaller error mean.

(c) The factors HL, SM, UL, W2, SS, OF, SP, and Ep were considered borderlines because it was not so clear to eliminate or to select the factor's level. Here, further investigation is needed.

(d) For the noise variables, statistical *t*-tests have not rejected the null hypothesis of equal means. As in design 1 the authors were quite comfortable in neglecting the tuning variables in

Table 4
2_{III}^{11-7} Design

	HL	SM	UL	W1	W2	SS	OF	SP	Ep	LR	ET	MAPE 1	MAPE 2	MAPE 3	Mean
1	−1	−1	−1	−1	−1	−1	−1	−1	1	1	1	0.060	0.073	0.068	0.067
2	1	−1	−1	−1	1	−1	1	1	−1	−1	−1	0.060	0.054	0.054	0.056
3	−1	1	−1	−1	1	1	−1	1	−1	−1	1	0.012	0.004	0.003	0.006
4	1	1	−1	−1	−1	1	1	−1	1	1	−1	0.099	0.097	0.092	0.096
5	−1	−1	1	−1	1	1	1	−1	−1	1	−1	0.072	0.074	0.074	0.074
6	1	−1	1	−1	−1	1	−1	1	1	−1	1	0.109	0.093	0.096	0.099
7	−1	1	1	−1	−1	−1	1	1	1	−1	−1	0.110	0.099	0.093	0.101
8	1	1	1	−1	1	−1	−1	−1	−1	1	1	0.056	0.055	0.053	0.055
9	−1	−1	−1	1	−1	1	1	1	−1	1	1	0.105	0.113	0.106	0.108
10	1	−1	−1	1	1	1	−1	−1	1	−1	−1	0.041	0.038	0.046	0.042
11	−1	1	−1	1	1	−1	1	−1	1	−1	1	0.027	0.025	0.024	0.025
12	1	1	−1	1	−1	−1	−1	1	−1	1	−1	0.105	0.112	0.114	0.110
13	−1	−1	1	1	1	−1	−1	1	1	1	−1	0.083	0.080	0.080	0.081
14	1	−1	1	1	−1	−1	1	−1	−1	−1	1	0.118	0.118	0.121	0.119
15	−1	1	1	1	−1	1	−1	−1	−1	−1	−1	0.080	0.097	0.087	0.088
16	1	1	1	1	1	1	1	1	1	1	1	0.093	0.095	0.105	0.098

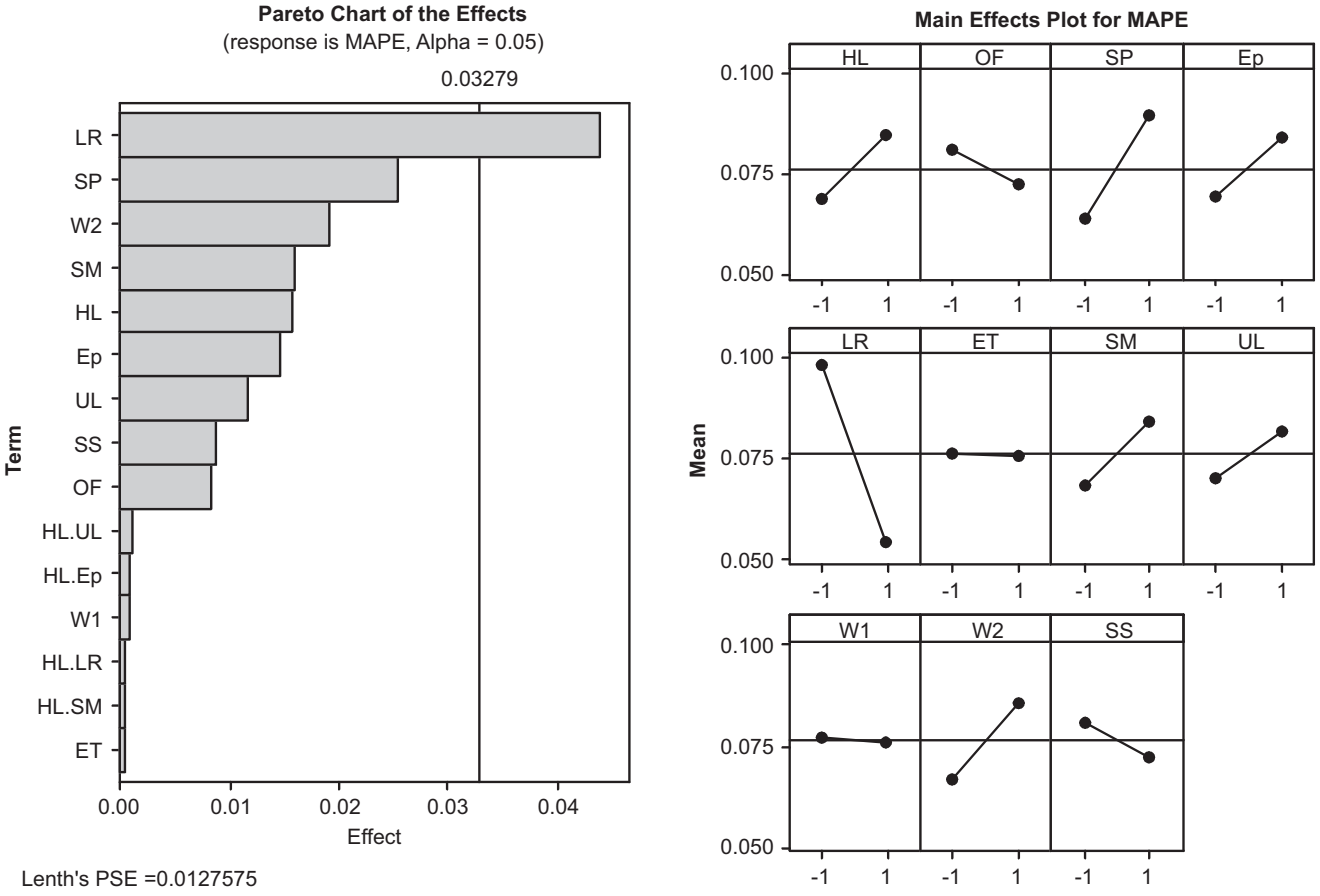

Fig. 5. 2_{III}^{11-7} Design (STAR2 model).

further designs. This mean that the setup obtained on previous setup, using Taguchi design, was coherent.

3.3.3. Design 3: 2^{8-4}_{IV} fractional factorial design for 8 factors

For the 8 two-level remaining factors a resolution L_{12} Taguchi design is the choice with minimum number of runs. Some close options could be the L_{16} Taguchi design or the resolution IV Fractional Factorial 2^{8-4}_{IV}design, both with16 runs. Due to interaction analysis, the choice of 2^{8-4}_{IV} is natural, considering the same criteria mentioned on previous design.

A resolution IV fractional factorial design has the main effects clear of two-factor interactions. Some two-factor interactions are aliased with each other. Resolution IV designs that contain exactly $2k$ runs are called *minimal designs*, as is in this case where $k = 8$.

The minimal design 2^{8-4}_{IV}results are shown in Table 5.

The *design generators* structure for this 2^{8-4}_{IV} minimal design is represented by

$$A = BCD, \quad F = ACD, \quad G = ABC, \quad H = ABD.$$

This means that a main factor (A, F, G and H) is confounded with three-factor interactions (BCD, ACD, ABC and ABD, respectively) that are usually of small effects. When analyzing the main effects, the greater the resolution the better.

Fig. 6 presents the main effects plot for the factors in Table 5. Some two-way interactions are also shown.

Similarly to design 2, this third design results were:

(a) The factor W2 was considered not significant. Its effect was constant for the factor's level.
(b) The factors SM, OF and Ep were considered significant and its levels could be established.
(c) The factors HL, UL, SS and SP were considered borderlines. Two-factor interactions were greater than some main factors but not statistically significant.
(d) For the noise variables, statistical *t*-tests have not rejected the null hypothesis of equal means.

3.3.4. Design 4: 2^4 full factorial design for 4 factors

For the 4 two-level remaining factors a full factorial design with 16 runs was chosen as shown in Table 6. The experimental results are shown in Fig. 7.

Table 5
2^{8-4}_{IV} Minimal design

	HL	SM	UL	W2	SS	OF	SP	Ep	MAPE 1	MAPE 2	MAPE 3	Mean
1	−1	−1	−1	−1	−1	−1	−1	−1	0.045	0.058	0.053	0.052
2	1	−1	−1	−1	1	−1	1	1	0.123	0.116	0.117	0.119
3	−1	1	−1	−1	1	1	−1	1	0.047	0.039	0.038	0.041
4	1	1	−1	−1	−1	1	1	−1	0.060	0.058	0.053	0.057
5	−1	−1	1	−1	1	1	1	−1	0.068	0.069	0.069	0.069
6	1	−1	1	−1	−1	1	−1	1	0.092	0.076	0.078	0.082
7	−1	1	1	−1	−1	−1	1	1	0.102	0.091	0.084	0.092
8	1	1	1	−1	1	−1	−1	−1	0.026	0.024	0.023	0.024
9	−1	−1	−1	1	−1	1	1	1	0.107	0.115	0.108	0.110
10	1	−1	−1	1	1	1	−1	−1	0.037	0.033	0.041	0.037
11	−1	1	−1	1	1	−1	1	−1	0.048	0.046	0.046	0.047
12	1	1	−1	1	−1	−1	−1	1	0.064	0.070	0.073	0.069
13	−1	−1	1	1	1	−1	−1	1	0.076	0.073	0.073	0.074
14	1	−1	1	1	−1	−1	1	−1	0.088	0.087	0.090	0.088
15	−1	1	1	1	−1	1	−1	−1	0.005	0.021	0.011	0.012
16	1	1	1	1	1	1	1	1	0.079	0.082	0.091	0.084

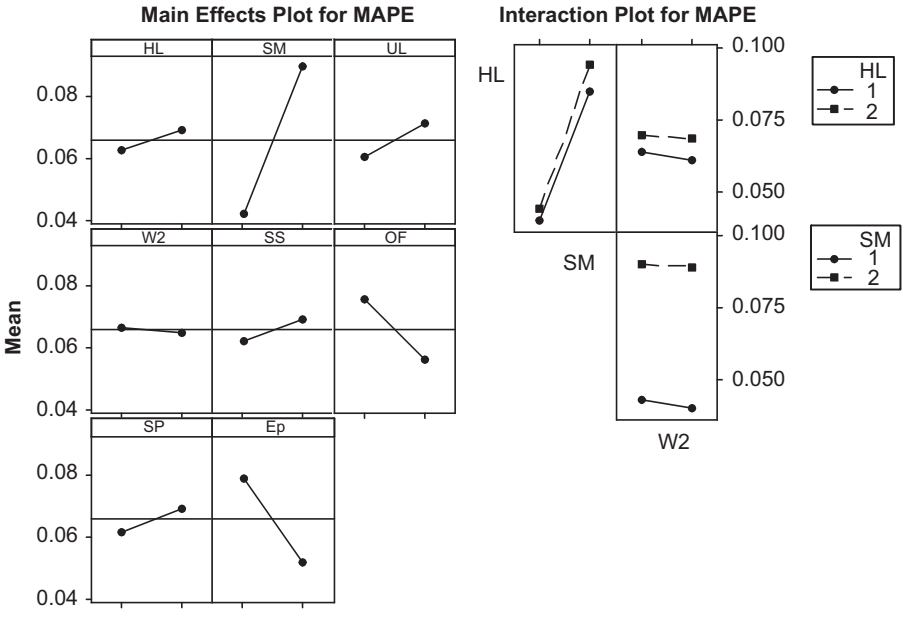

Fig. 6. 2^{8-4}_{IV} Design (STAR2 model).

For this full factorial design, the results were:

(a) The factors HL, UL, SS and SP were considered significant and its levels could be established.
(b) A second-order interaction between HL and UL was considered significant.
(c) For the noise variables, statistical *t*-tests have not rejected the null hypothesis of equal means.

Table 6
Full factorial design

	HL	UL	SS	SP	MAPE 1	MAPE 2	MAPE 3	Mean
1	−1	−1	−1	−1	0.10	0.12	0.11	0.11
2	1	−1	−1	−1	0.07	0.06	0.06	0.07
3	−1	1	−1	−1	0.07	0.06	0.06	0.06
4	1	1	−1	−1	0.05	0.05	0.05	0.05
5	−1	−1	1	−1	0.07	0.07	0.07	0.07
6	1	−1	1	−1	0.04	0.02	0.02	0.03
7	−1	1	1	−1	0.03	0.02	0.01	0.02
8	1	1	1	−1	0.02	0.01	0.01	0.01
9	−1	−1	−1	1	0.12	0.12	0.12	0.12
10	1	−1	−1	1	0.07	0.07	0.08	0.07
11	−1	1	−1	1	0.07	0.07	0.07	0.07
12	1	1	−1	1	0.06	0.06	0.07	0.06
13	−1	−1	1	1	0.08	0.08	0.08	0.08
14	1	−1	1	1	0.03	0.03	0.04	0.03
15	−1	1	1	1	0.02	0.04	0.03	0.03
16	1	1	1	1	0.02	0.02	0.03	0.02

3.4. Confirmation tests

Table 7 presents the simulation results for the eight nonlinear time series. The designs used on each nonlinear model are also mentioned. They were obtained according to similar procedure as shown in previous section. The final ANN's MAPE and the model's MAPE were also computed.

The main findings on these experimental results are following summarized.

- The factors HL, SM, UL, SS, OF, P1, P2, PT were considered significant for the eight nonlinear time series. The features in each nonlinear model were irrelevant determining the ANN parameters.
- For most time series the *Sampling method* (SM) interacts significantly with *Number of units per layer* (UL). This is theoretically expected since complex Neural Networks (like with four layers, e.g.) requires more samples for training. The same expectation happens with the HL and UL. For interaction between UL and HL the main factors act in the way to decrease the MAPE and the interaction acts on the opposite way.

Fig. 8 shows the ANN forecasting results for the SETAR2 model using the optimized parameters as presented in Table 6. The ANN presents a good fitting when compared to the specific nonlinear model.

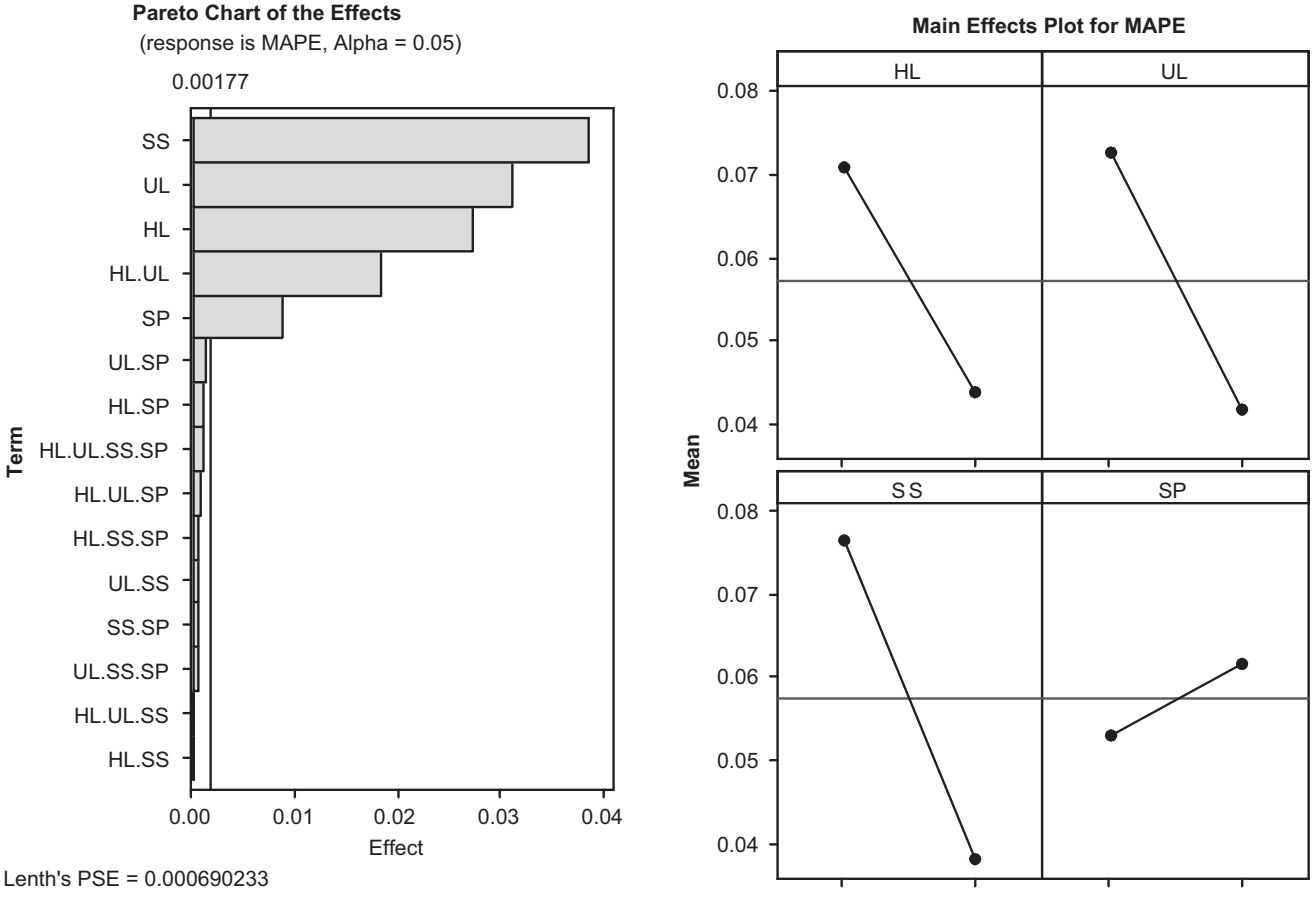

Fig. 7. 2^4 Design (STAR2 model).

Table 7
ANN parameters chosen for the nonlinear time series

Factor	Nonlinear model							
	SAR	BL1	BL2	TAR	NAR	NMA	STAR1	STAR2
HL	2**	2*	2*	2*	2*	2*	2**	2**
SM	Random*	Random**	Random*	Random*	Random*	Random*	Random**	Random**
UL	2(*N*+1)**	2(*N*+1)*	2(*N*+1)*	2(*N*+1)**	2(*N*+1)*	2(*N*+1)*	2(*N*+1)**	2(*N*+1)**
W1	0.0001	0.0001	0.0001	0.001	0.001	No	0.001	0.001
W2	0.0001	No	No	0.001	0.0001	No	0.001	0.001
SS	21 × 24*	21 × 24*	21 × 24*	21 × 24*	21 × 24*	21 × 24*	21 × 24**	21 × 24**
OF	Logistic*	Logistic*	Logistic*	Logistic*	Logistic*	Logistic*	Logistic*	Logistic*
SP	24	12*	12	12*	24	12*	24	12*
Ep	400	400	400*	100	100	400	400*	400*
LR	0.9*	0.9	0.9	0.9*	0.9*	0.1	0.9*	0.9*
ET	0	−0.1	−0.1	0	0	−0.1	0	0
P1	BP*	BP*	BP*	BP*	BP*	BP*	BP*	BP*
P2	Lev. Mar.*	Lev. Mar.**	Lev. Mar.**	Lev. Mar.*	Lev. Mar.*	Lev. Mar.*	Lev. Mar.*	Lev. Mar.**
PT	Univ.**	Univ.**	Univ.**	Univ.**	Univ.**	Univ.**	Univ.**	Univ.**
PU	Sensitivity	No	Fan-out	Fan-out	Sensitivity	No	No	Fan-out
EE	25	1	1	50	25	50	25	25
PI	No	No	Fan-out	Fan-out	Sensitivity	Sensitivity	Fan-out	Fan-out
IM	*U*(0,1)	*U*(0,1)	*U*(0,1)	*N*(0,1)	*N*(0,1)	*U*(0,1)	*N*(0,1)	*N*(0,1)
SC	0	0.1	0.1	0	0.1	0	0	0
BP	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On
QP	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On
DD	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On
Main interactions	HL.UL SM.UL Tag L_{36} L_4	HL.UL OF.P2 SM.UL Tag L_{36} L_4	HL.UL SM.UL Tag L_{36} L_4	SM.UL Tag L_{36} L_4	HL.UL OF.P2 SM.UL Tag L_{36} L_4	HL.UL P2.PT Tag L_{36} L_4	HL.UL SM.UL Tag L_{36} L_4	HL.UL Tag L_{36} L_4
Factorial designs	2_{III}^{11-7} 2_{IV}^{7-3} 2^3	2_{III}^{10-6} 2_{IV}^{6-2} 2^3	2_{III}^{10-6} 2_{IV}^{7-3} 2^4	2_{III}^{9-5} 2_{IV}^{6-2} 2^3	2_{III}^{11-7} 2_{IV}^{8-4} 2^4	2_{III}^{11-7} 2_{IV}^{7-3} 2^3	2_{III}^{12-8} 2_{IV}^{8-4} 2_{V}^{5-1}	2_{III}^{11-7} 2_{IV}^{8-4} 2^4
MAPE (ANN)	0.0435	0.0292	0.0194	0.0248	0.0366	0.0468	0.0246	0.0282
MAPE (Model)	0.0623	0.061	0.0656	0.0781	0.0664	0.0789	0.075	0.0788

* *P*-value < 0.05.
** *P*-value < 0.001.

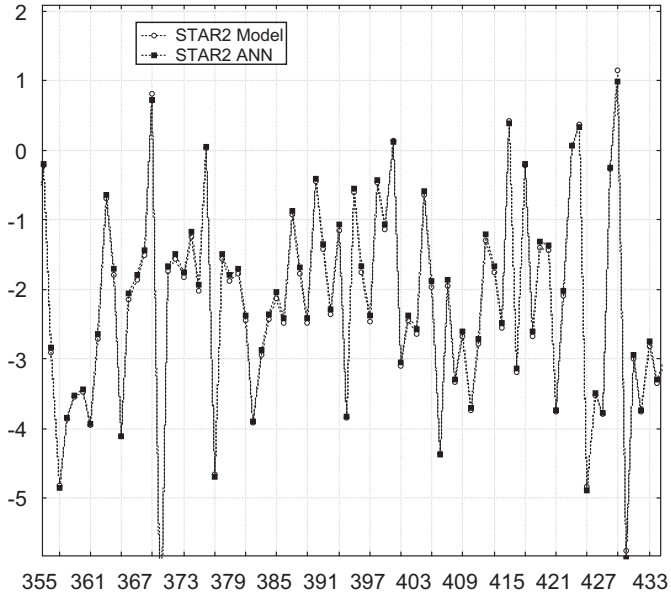

Fig. 8. ANN forecasting for the SETAR2 model using the optimized parameters. The data test is amplified.

4. Case study: short-term electricity load

Companies that trade in deregulated electricity markets in the US, Brazil, England, and most other countries use time series forecasting to predict demand of electricity as part of the information necessary to set buying and selling contracts. If they are production or trade companies, their interests lie particularly in optimizing the energy load and prices sold to their costumers.

This case study deals with the modeling and forecasting of an important variable that affects achieving a good portfolio of electricity contracts: the electricity load (or consumption). Knowledge of its future value and its variation is essential to calculate the portfolio risk and return. Although it is possible to contract a certain amount of energy with no allowed variation, for a consumer it represents high risk. In the industrial process there are a lot of unpredictable factors that can cause an increase or decrease in energy consumption. If an electricity producer can accept and manage this risk, the producer will have more opportunities in the market. Thus, it is important to manage these risks.

As considered by Angelus [2] and Mount [41], forecasting electricity prices and loads is an arduous task due to a great number of factors such as electricity demand and supply, number of generations, transmission and distribution constraints, etc. For

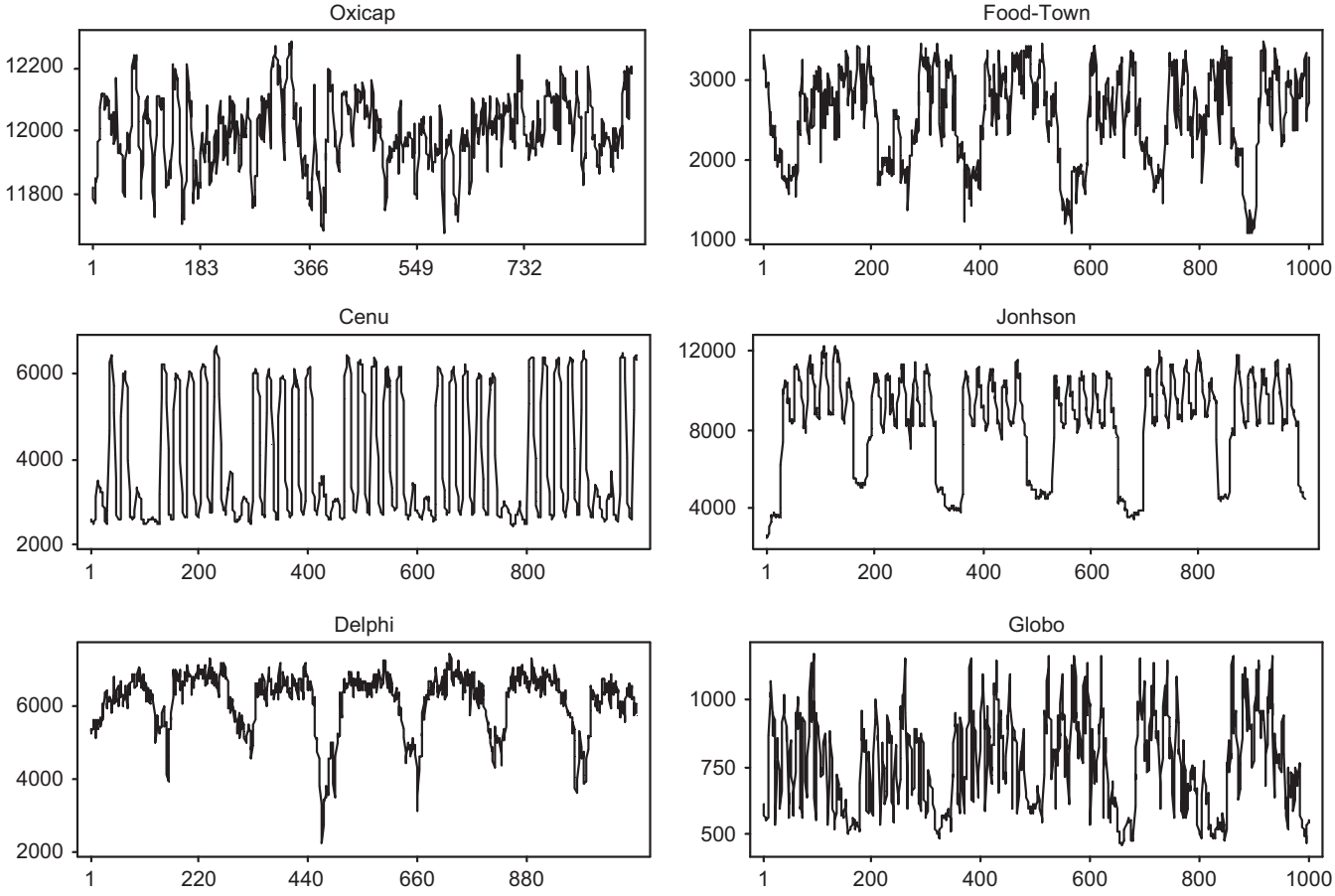

Fig. 9. Hourly-based time series (in Watts) of six industrial consumers.

deregulated electricity markets, a great volume of electricity is bought and sold in the spot market as well as in the bilateral market among agents at different geographical regions. A simple unpredictable climate change can add volatility in demand and therefore volatility in prices. Pai and Hong [44] also state that electricity load forecasting is complex to conduct due to nonlinearity of its influenced factors. They state that the most important factor in regional or national power system strategy management has been accurate load forecasting of future demand.

This case study intends to use the DOE/ANN framework to forecast electricity load of six industrial consumers in Brazil using historical hourly time series.

Fig. 9 represents the time series pattern of the hourly electricity load for each of the six industrial consumers. Procedures to improve data collection were here enforced to treat missing data, outliers, typos, seasonality and errors. Week seasonality was observed in all time series (of lag 168). The available data set comprises 3 years of hourly electricity load of Duke Energy, a production company that operates in Brazil.

One problem that comes up when dealing with multiple time series is usually related to the way the forecasting method is handled. In this regard it is worth mentioning that for this entire set of electricity load the behavior of the six time series cannot be considered as a multivariate process. The correlation between variables was not significant in any pair of the six time series. Also, some factors such as missing data, different number of samples and different intervals, made this kind of approach less likely to occur. In this way, each series needs to be independently modeled. Some time series that come from industrial customers with the same activity could be explained by the same variables, but this fact was not considered in this paper. Only one multivariate specific method with the same parameters is not usually able to fit the whole set of time series.

Table 8 presents the main results for the ANN's training and electricity load forecasting for the six time series considered. Some nonlinear models were also estimated for comparative purposes and the MAPE error was obtained considering $T = 24$ h (1-day ahead) as interval of prediction. MAPE results are statistically significant when compared to the nonlinear models adopted for each time series. It is correct to assume that the ANNs were quite competent on predicting the short-term electricity load for the industrial consumers.

5. Conclusions and further research

Motivated by the lack of evidence that an ANN can be easily parameterized, this study applied the methodology of DOE to optimize the ANN's training for problems of nonlinear time series. The paper describes an approach based on factorial DOE using screening, Taguchi, fractional and full factorial designs to set the parameters of a feedforward multilayer perceptron neural network. The approach uses classical factorial designs to sequentially define the main ANN parameters that a minimum prediction error could be reached.

The main factors and interactions were identified using this approach and results suggest that ANNs using DOE can perform better comparably to the existent nonlinear autoregressive models.

Eight synthetic nonlinear time series were estimated using ANN and results have shown that the ANNs were able to

Table 8
ANN parameters for the six industrial electricity load consumer

Factor	Electricity load					
	Cenu	Food-Town	Oxicap	Delphi	Johnson	Globo
HL	2*	2**	2*	2**	2**	2**
SM	Random**	Random**	Random*	Random*	Random*	Random**
UL	50**	50*	50*	50**	50*	50*
W1	No	0.001	0.001	No	0.001	0.001
W2	No	0.001	0.01	No	0.0001	0.01
SS	21×24**	21×24*	21×24**	21×24*	21×24*	21×24**
OF	Logistic**	Logistic**	Logistic**	Logistic*	Logistic*	Logistic**
SP	48	24*	24	48*	24	24*
Ep	500	500	400*	400	500	400
LR	0.9*	0.9	0.9	0.9*	0.9*	0.9
ET	−0.1	−0.1	0	0	0	0
P1	BP**	BP*	BP*	BP**	BP*	BP**
P2	Lev. Mar.*	Lev. Mar.**	Lev. Mar.**	Lev. Mar.*	Lev. Mar.*	Lev. Mar.*
PT	Univ.*	Univ.*	Univ.**	Univ.**	Univ.**	Univ.*
PU	Fan-out	Fan-out	Fan-out	Sensitivity	Fan-out	Fan-out
EE	1	25	25	25	50	25
PI	Fan-out	Sensitivity	Fan-out	Sensitivity	Fan-out	Fan-out
IM	$N(0,1)$	$U(0,1)$	$N(0,1)$	$N(0,1)$	$N(0,1)$	$U(0,1)$
SC	0.1	0.1	0.1	0.1	0	0
BP	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On
QP	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On
DD	Factors On	Factors On	Factors On	Factors On	Factors On	Factors On
Main interactions	HL.UL P2.PT Tag L_{36} L_4	SM.UL Tag L_{36} L_4	HL.UL OF.P2 SM.UL Tag L_{36} L_4	HL.UL SM.UL Tag L_{36} L_4	HL.UL OF.P2 SM.UL Tag L_{36} L_4	HL.UL OF.P2 SM.UL Tag L_{36} L_4
Factorial designs	2_{III}^{10-6} 2_{IV}^{7-3} 2^4	2_{III}^{11-7} 2_{IV}^{8-4} 2^4	2_{III}^{9-5} 2_{IV}^{6-2} 2^3	2_{III}^{10-6} 2_{IV}^{7-3} 2^4	2_{III}^{10-6} 2_{IV}^{6-2} 2^3	2_{III}^{11-7} 2_{IV}^{7-3} 2^3
MAPE (ANN)	0.102	0.053	0.113	0.095	0.872	0.105
MAPE (Model)	0.186 (STAR)	0.101 (BILINEAR))	0.173 (BILINEAR)	0.163 (STAR)	0.107 (NAR)	0.173 (BILINEAR)

* P-value <0.05.
** P-value <0.001.

generalize realistic autoregressive models better than the specific model proposed. This general nonlinear property makes ANNs a promising alternative when forecasting these models. Six industrial electricity load time series were explored as a case study and results have confirmed the proposed DOE/ANN framework.

As a further research, the following points need to be better addressed:

- The evolutionary approach (e.g., genetic algorithm) have been referred as an exception in terms of training an ANN, in which such evolution levels as connection weights, network structure, learning rules, etc., are optimized using an evolutionary algorithm. However, studies on the evolutionary approach are usually concerned with a certain level of evolution, and research into the simultaneous evolution of various levels is still in its early stage [59].
- Several different methods have been proposed for building the optimal architecture of an ANN and consider topological factors such as the number of inputs, the number of hidden layers and the number of neurons. The pruning algorithm [47], the polynomial time algorithm [48], the canonical decomposition technique [58], and the network information criterion [42] are among these methods. None of them, however, can guarantee the best ANN solution for the nonlinear time series forecasting problem. One interesting research is to consider these methods also as variable in the present DOE framework.

Acknowledgments

The authors thank the Brazilian governmental agencies of CAPES and FAPEMIG for supporting this research.

References

[1] M.J. Aitkenhead, A.J.S. McDonald, J.J. Dawson, G. Couper, R.P. Smart, M. Billett, D. Hope, S. Palmer, A novel method for training neural networks for time-series prediction in environmental systems, Ecol. Model. 162 (2003) 87–95.
[2] A. Angelus, Electricity price forecasting in deregulated markets, Electricity J. 4 (2001) 32–41.
[3] S.D. Balkin, J.K. Ord, Automatic neural network modeling for univariate time series, Int. J. Forecast. 16 (2000) 509–515.
[4] C. Bishop, Neural Networks for Pattern Recognition, University Press, Oxford, 1995.
[5] G.E.P. Box, G.M. Jenkins, Time Series Analysis: Forecasting and Control, revised ed., Holden-Day, San Francisco, 1976.
[7] S. BuHamra, N. Smaoui, M. Gabr, The Box–Jenkins analysis and neural networks: prediction and time series modeling, Appl. Math. Model. 27 (2003) 805–815.
[8] K.K. Chan, T.A. Spedding, On-line optimization of quality in a manufacturing system, Int. J. Prod. Res. 39 (6) (2001) 1127–1145.
[9] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (1995) 911–917.
[10] R. Chen, R.S. Tsay, Functional-coefficient autoregressive models, J. Am. Stat. Assoc. 88 (1993) 298–308.
[11] B. Cheng, D.M. Titterington, Neural networks: a review from a statistical perspective, Stat. Sci. 9 (1994) 2–54.
[12] W.C. Chiang, T.L. Urban, G.W. Baldridge, A neural network approach to mutual fund net asset value forecasting, Omega, Int. J. Manage. Sci. 24 (1996) 205–215.

[13] J.G. de Gooijer, On threshold moving-average models, J. Time Ser. Anal. 19 (1998) 1–18.
[14] F.X. Diebold, J.A. Nason, Nonparametric exchange rate prediction, J. Int. Econ. 28 (1990) 15–332.
[16] E.U. Enemuoh, A.S. El-Gizawy, Optimal neural network model for characterization of process-induced damage in drilling carbon fiber reinforced epoxy composites, Mach. Sci. Technol. 7 (3) (2003) 389–400.
[17] P.H. Franses, P. van Homelen, On forecasting exchange rates using neural networks, Appl. Financ. Econ. 8 (1998) 689–696.
[18] M.C. Fu, Optimization for simulation: theory vs. practice, INFORMS J. Comput. 14 (2002) 192–215.
[19] M. Ghiassi, H. Saidane, D.K. Zimbra, A dynamic artificial neural network model for forecasting time series events, Int. J. Forecast. 21 (2005) 341–362.
[20] C.W.J. Granger, A.P. Anderson, An Introduction to Bilinear Time Series Models, Vandenhoeck & Ruprecht, Gottingen, 1978.
[21] C.W.J. Granger, T. Terasvirta, Modelling Nonlinear Economic Relationships, Oxford University Press, 1993.
[22] J.D. Hamilton, A new approach to the economic analysis of nonstationary time series and the business cycle, Econometrica 57 (1989) 357–384.
[23] S.L. Ho, M. Xie, T.N. Goh, A comparative study of neural network and Box–Jenkins ARIMA modeling in time series prediction, Comput. Ind. Eng. 42 (2002) 371–375.
[24] K. Hornik, Some new results on neural network approximation, Neural Netw. 6 (1993) 1069–1072.
[25] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[26] H.B. Hwarng, Insights into neural-network forecasting of time series corresponding to ARMA(p; q) structures, Omega 29 (2001) 273–289.
[27] H.B. Hwarng, H.T. Ang, A simple neural network for ARMA(p; q) time series, Omega 29 (2001) 319–333.
[28] I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series, Neurocomputing 10 (1996) 215–236.
[29] K. Kalaitzakis, G.S. Stavrakakis, E.M. Anagnostakis, Short-term load forecasting based on artificial neural networks parallel implementation, Electric Power Syst. Res. 63 (2002) 185–196.
[30] D.S.K. Karunasinghe, S.Y. Liong, Chaotic time series prediction with a global model: artificial neural network, J. Hydrol. 2 (2005) 1–14.
[31] J.F.C. Khaw, B.S. Lim, L.E.N. Lim, Optimal design of neural networks using the Taguchi method, Neurocomputing 7 (1995) 225–245.
[32] K.J. Kim, Financial time series forecasting using support vector machines, Neurocomputing 55 (2003) 307–319.
[33] Y.S. Kim, B.J. Yum, Robust design of multilayer feedforward neural networks: an experimental approach, Eng. Appl. Artif. Intell. 17 (2004) 249–263.
[34] J.P.C. Kleijnen, S.M. Sanchez, T.W. Lucas, T.M. Cioppa, State-of-the-art review: a user's guide to the brave new world of designing simulation experiments, INFORMS J. Comput. 7 (3) (2005) 263–289.
[35] N. Kohzadi, M.S. Boyd, B. Kermanshahi, I. Kaastra, A comparison of artificial neural network and time series models for forecasting commodity prices, Neurocomputing 10 (1996) 169–181.
[36] H. Krager, P. Kugle, Nonlinearities in foreign exchange markets: a different perspective, J. Int. Money Finance 12 (1993) 195–208.
[37] T.Y. Lin, C.H. Tseng, Optimum design for artificial neural networks: an example in a bicycle derailleur system, Eng. Appl. Artif. Intell. 13 (2000) 3–14.
[38] S. Makridakis, M. Hibon, The M3-competition: results, conclusions and implications, Int. J. Forecast. 16 (2000) 451–476.
[39] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, R. Winkler, The accuracy of extrapolation (time series) methods: results of a forecasting competition, J. Forecast. 1 (1982) 111–153.
[40] S. Makridakis, C. Chatfield, M. Hibon, M. Lawrence, T. Mills, K. Ord, L.F. Simmons, The M-2 competition: a real-time judgmentally based forecasting study, Int. J. Forecast. 9 (1993) 5–23.
[41] T. Mount, Market power and price volatility in restructured markets for electricity, Decis. Support Syst. 30 (2001) 311–325.
[42] N. Murata, S. Yoshizawa, S. Amari, Network information criterion-determining the number of hidden units for an artificial neural network model, IEEE Trans. Neural Netw. 5 (1994) 865–872.
[43] H. Niska, T. Hiltunen, A. Karppinen, J. Ruuskanen, M. Kolehmainen, Evolving the neural network model for forecasting air pollution time series, Eng. Appl. Artif. Intell. 17 (2004) 159–167.
[44] P.F. Pai, W.C. Hong, Support vector machines with simulated annealing algorithms in electricity load forecasting, Energy Convers. Manage. 46 (2005) 2669–2688.
[45] M.B. Priestley, State-dependent models: a general approach to nonlinear time series analysis, J. Time Ser. Anal. 1 (1980) 47–71.
[46] M. Qi, G.P. Zhang, An investigation of model selection criteria for neural network time series forecasting, Eur. J. Operat. Res. 132 (2001) 666–680.
[47] R. Reed, Pruning algorithms- a survey, IEEE Trans. Neural Netw. 4 (1993) 740–747.
[48] A. Roy, L.S. Kim, S. Mukhopadhyay, A polynomial time algorithm for the construction and training of a class of multilayer perceptrons, Neural Netw. 6 (1993) 535–545.
[49] Z. Shi, Y. Tamura, T. Ozaki, Nonlinear time series modelling with the radial basis function-based state-dependent autoregressive model, Int. J. Syst. Sci. 30 (7) (1999) 717–727.
[50] J.C. Spall, Introduction to Stochastic Search and Optimization; Estimation, Simulation, and Control, Wiley, New York, 2003.
[51] StatSoft Inc., Statistica (data analysis software system), version 7.1, 2005, 〈www.statsoft.com〉.
[52] W. Sukthomya, J. Tannock, The optimisation of neural network parameters using Taguchi's design of experiments approach: an application in manufacturing process modeling, Neural Comput. Appl. 14 (2005) 337–344.
[53] W. Sukthomya, J. Tannock, The training of neural networks to model manufacturing processes, J. Intell. Manuf. 16 (2005) 39–51.
[54] T. Terasvirta, D. van Dijk, M.C. Medeiros, Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: a re-examination, Int. J. Forecast. 21 (2005) 755–774.
[55] H. Tong, On a threshold model, in: C.H. Chen (Ed.), Pattern Recognition and Signal Processing, Sijhoff & Noordhoff, Amsterdam, 1978.
[56] R. Tsay, Analysis of Financial Time Series, second ed., Wiley-Interscience, 2005.
[57] F.M. Tseng, H.C. Yu, G.H. Tzeng, Combining neural network model with seasonal time series ARIMA model, Technol. Forecast. Soc. Change 72 (69) (2002) 71–87.
[58] Z. Wang, C.D. Massimo, M.T. Tham, A.J. Morris, A procedure for determining the topology of multilayer feedforward neural networks, Neural Netw. 7 (1994) 291–300.
[59] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (9) (1999) 1423–1447.
[60] G.P. Zhang, An investigation of neural networks for linear time-series forecasting, Comput. Operat. Res. 28 (2001) 1183–1202.
[61] G.P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175.
[62] G.P. Zhang, M. Qi, Neural network forecasting for seasonal and trend time series, Eur. J. Operat. Res. 160 (2005) 501–514.
[63] G.P. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with artificial neural networks: the state-of-the-art, Int. J. Forecast. 14 (1998) 35–62.
[64] G.P. Zhang, B.E. Patuwo, M.Y. Hu, A simulation study of artificial neural networks for nonlinear time-series forecasting, Comput. Operat. Res. 28 (2001) 381–396.
[65] G. Taguchi, Systems of Experimental Design, vols. 1 & 2, UNIPUB/Kraus International Publications, New York, 1987.
[66] Y. Bodyanskiy, S. Popov, Neural network approach to forecasting of quasiperiodic financial time series, Eur. J. Oper. Res. 175 (3) (2006) 1357–1366.
[67] M.-D. Cubiles-de-la-Vega, R.P. Mejıas, A.P. Acosta, J.M. Garcıa, Intell. Data Anal. 6 (2002) 53–65.
[68] D.E. Coleman, D.C. Montgomery, A systematic approach to planning for a designed industrial experiment Technometrics 35 (1993) 1–27.
[69] G. Derringer, R. Suich, Simultaneous optimization of several response variables, J. Qual. Technol. 12 (4) (1980) 214–219.
[70] Crary Group, 2004, http://www.webdoe.cc/.
[71] J.S. Armstrong, F. Collopy, Error measures for generalizing about forecasting methods: empirical comparisons, Int. J. Forecast 8 (1992) 99–111.
[72] R.H. Myers, D.C. Montgomery, Response Surface Methodology: Process and Product Optimization Using Designed Experiments, Wiley, New York, 1995.
[73] J.P.C. Kleijnen, Statistical Tools for Simulation Practitioners., Marcel Dekker Inc. Publ., New York, 1987.
[74] L. Trocine, L.C. Malone, A.L. Trocine, L.C. Malone, Finding important independent variables through screening designs: a comparison of methods, in: Proceedings of the Winter Simulation Conference, 2000.
[75] T.H. Andres, W.C. Hajas, Using iterated fractional factorial design to screen parameters in sensitivity analysis of a probabilistic risk assessment model, in: Proceedings of the Joint International Conference on Mathematical Models and Supercomputing in Nuclear Applications, Karlsruhe, Germany, 1993, pp. 19–23.
[76] L. Trocine, An efficient, effective, and robust procedure for screening more than 20 independent variables employing a genetic algorithm, Dissertation, University of Central Florida, Orlando, FL, USA, 2001.

Pedro P. Balestrassi received the D.Sc. degree in Industrial Engineering from the Federal University of Santa Catarina, Brazil, in 2000. Since 1994, he has been with the Federal University at Itajuba, Brazil, as Professor of Industrial Engineering. In 1998/1999, he was visitor at Texas A&M University (TX, USA) and in 2005/2006 he was visiting the University of Texas at Austin. His areas of interest lie in times series forecasting, ANN and statistics.

Elmira Popova received the M.Sc. degree in Mathematics in 1985 from the University of Sofia, Bulgaria and the Ph.D. degree in Operations Research in 1995 from Case Western Reserve University, Cleveland, OH, USA. Dr. Popova specializes in stochastic processes and statistics. Her research interests include reliability analysis and design of optimal maintenance strategies for randomly failing systems. Dr. Popova is associate professor of Operations Research/Industrial Engineering at University of Texas at Austin.

Anderson Paulo de Paiva is Professor of Quantitative Methods in the Industrial Engineering Institute of the Federal University at Itajuba, Brazil. He received the D.Sc. degree in Industrial Engineering in Mechanical Engineering in 2006 at the Federal University of Itajuba, Brazil. His research interests are in Multivariate Statistical Analysis, Nonlinear Optimization and Neural Networks applied to manufacturing processes.

José W. Marangon Lima received the B.Sc., M.Sc. and D.Sc. degrees in electrical engineering, the latter from Federal University of Rio de Janeiro (COPPE-UFRJ), Rio de Janeiro, Brazil, in 1994. Since 1993, he has been with the Federal University at Itajuba, Brazil, as a Professor of Electrical Engineering. He was visiting professor at University of Texas at Austin in 2005/2006.